Lecture Notes in Artificial Intelligence 5360

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Wayne Wobcke Mengjie Zhang (Eds.)

AI 2008: Advances in Artificial Intelligence

21st Australasian Joint Conference on Artificial Intelligence
Auckland, New Zealand, December 1-5, 2008
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Wayne Wobcke
University of New South Wales
School of Computer Science and Engineering
Sydney NSW 2052, Australia
E-mail: wobcke@cse.unsw.edu.au

Mengjie Zhang
Victoria University of Wellington
School of Mathematics, Statistics and Computer Science
P.O. Box 600, Wellington 6140, New Zealand
E-mail: mengjie.zhang@mcs.vuw.ac.nz

Library of Congress Control Number: 2008938719

CR Subject Classification (1998): I.2, F.4.1, H.3, H.2.8, F.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-89377-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-89377-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12566891 06/3180 5 4 3 2 1 0

Preface

AI 2008, the 21st Australasian Joint Conference on Artificial Intelligence, was, for the first time, held in New Zealand, in Auckland during December 1–5, 2008. The conference was hosted by Auckland University of Technology.

AI 2008 attracted 143 submissions from 22 countries, of which 42 (29%) were accepted as full papers and 21 (15%) as short papers. Submissions were subject to a rigorous review process. Each paper was reviewed by at least three (often four, and in one case, six) members of the Programme Committee. Authors could then provide a "rebuttal" to these reviews. The Senior Programme Committee members coordinated discussion on the papers to provide a recommendation of acceptance or rejection to the Programme Committee Co-chairs. Both full papers and short papers were presented at the conference.

We would first like to thank all those who submitted papers to AI 2008. Special thanks to the Programme Committee members for their detailed reviews completed in a timely manner, and to the Senior Programme Committee for their considered judgements and recommendations on the papers. We are sure authors would like to know that the rebuttal and subsequent discussion phases made a difference to the outcome in numerous cases. We are confident that this process has improved the decision making for final paper selection, and that the overall quality and reputation of the conference is enhanced as a result. Thanks also to EasyChair for the use of their conference management system to facilitate this complex process and the preparation of these proceedings.

AI 2008 featured three invited talks, from Tony Cohn ("Steps Towards Cognitive Vision"), Reinhard Klette ("Stereo-Vision-Based Driver Assistance") and Zbigniew Michalewicz ("Intelligence, Business Intelligence, and Adaptive Business Intelligence"). These talks contributed greatly to the intellectual environment of the conference, and were highly appreciated by all participants.

Being the first "Australasian" conference continuing the series of Australian conferences, this year was somewhat of an experiment. We would like to acknowledge the large number of New Zealand researchers who submitted papers and served on the Programme Committee of AI 2008, helping to make this conference a success. We would like to thank Auckland University of Technology for organizing the conference, and the Australian Computer Society, the University of New South Wales and the University of Wollongong for financial support.

December 2008

Wayne Wobcke
Mengjie Zhang

Organization

AI 2008 was hosted by Auckland University of Technology, and was held in Auckland, New Zealand, during December 1–5, 2008.

Conference Committee

General Co-chairs

Wai-Kiang (Albert) Yeap	Auckland University of Technology, New Zealand
Aditya Ghose	University of Wollongong, Australia

Programme Committee Co-chairs

Wayne Wobcke	University of New South Wales, Australia
Mengjie Zhang	Victoria University of Wellington, New Zealand

Local Arrangements Chair

Yun Sing Koh	Auckland University of Technology, New Zealand

Senior Programme Committee

Hussein Abbass	ADFA, University of New South Wales, Australia
Stephen Cranefield	University of Otago, New Zealand
Robert Dale	Macquarie University, Australia
James Delgrande	Simon Fraser University, Canada
David Dowe	Monash University, Australia
Achim Hoffmann	University of New South Wales, Australia
Byeong Ho Kang	University of Tasmania, Australia
Reinhard Klette	University of Auckland, New Zealand
Jérôme Lang	IRIT, France
Xiaodong Li	RMIT University, Australia
John Lloyd	Australian National University, Australia
Brendan McCane	University of Otago, New Zealand
Detlef Nauck	British Telecom, UK
Mehmet Orgun	Macquarie University, Australia
Maurice Pagnucco	University of New South Wales, Australia
Abdul Sattar	Griffith University, Australia
John Thornton	Griffith University, Australia

Toby Walsh	National ICT Australia, Australia
Xindong Wu	University of Vermont, USA
Xinghuo Yu	RMIT University, Australia
Chengqi Zhang	University of Technology, Sydney, Australia
Yan Zhang	University of Western Sydney, Australia
Zhi-Hua Zhou	Nanjing University, China

Programme Committee

Peter Andreae (New Zealand)
Yun Bai (Australia)
James Bailey (Australia)
Michael Bain (Australia)
Tim Baldwin (Australia)
Roberto Battiti (Italy)
Peter Baumgartner (Australia)
Lubica Benuskova (New Zealand)
Michael Berthold (Germany)
Ghassan Beydoun (Australia)
Richard Booth (Thailand)
Rafael Bordini (UK)
Adi Botea (Australia)
Sebastian Brand (Australia)
Thomas Bräunl (Australia)
Wray Buntine (Australia)
Jinhai Cai (Australia)
Longbing Cao (Australia)
Lawrence Cavedon (Australia)
Chia-Yen Chen (Taiwan)
Songcan Chen (China)
Andrew Chiou (Australia)
Sung-Bae Cho (South Korea)
Grace Chung (Australia)
Vic Ciesielski (Australia)
Dan Corbett (USA)
Michael Cree (New Zealand)
Hepu Deng (Australia)
Jeremiah Deng (New Zealand)
Grant Dick (New Zealand)
Yulin Ding (Australia)
Xiangjun Dong (China)
Mark Dras (Australia)
Atilla Elçi (Turkey)
Esra Erdem (Turkey)
Daryl Essam (Australia)
Ling Feng (China)
Cèsar Ferri (Spain)
Marcus Frean (New Zealand)
Alfredo Gabaldon (Australia)
Yang Gao (China)
Xin Geng (China)
Manolis Gergatsoulis (Greece)
Guido Governatori (Australia)
Charles Gretton (Australia)
Hans Guesgen (New Zealand)
Fikret Gürgen (Turkey)
Patrik Haslum (Australia)
Bernhard Hengst (Australia)
José Hernández-Orallo (Spain)
Philip Hingston (Australia)
Geoffrey Holmes (New Zealand)
Wei-Chiang Hong (Taiwan)
Marcus Hutter (Australia)
Licheng Jiao (China)
Warren Jin (Australia)
Zhi Jin (China)
Ken Kaneiwa (Japan)
George Katsirelos (Australia)
Paul Kennedy (Australia)
Philip Kilby (Australia)
Frank Klawonn (Germany)
Alistair Knott (New Zealand)
Mario Köppen (Japan)
Rudolf Kruse (Germany)
Rex Kwok (Australia)
Willem Labuschagne (New Zealand)
Gerhard Lakemeyer (Germany)
Jimmy Lee (Hong Kong)
Maria R. Lee (Taiwan)
Yves Lespérance (Canada)
Bin Li (China)

Li Li (Australia)
Wei Li (Australia)
Xiao-Lin Li (China)
Yuefeng Li (Australia)
Jing Liu (China)
Xudong Luo (UK)
Bruce MacDonald (New Zealand)
Eric Martin (Australia)
Rodrigo Martínez-Béjar (Spain)
Kathryn Merrick (Australia)
Thomas Meyer (South Africa)
Diego Mollá (Australia)
John Morris (New Zealand)
Saeid Nahavandi (Australia)
Richi Nayak (Australia)
David Newth (Australia)
Kouzou Ohara (Japan)
Cécile Paris (Australia)
Francis Jeffry Pelletier (Canada)
Pavlos Peppas (Greece)
Nicolai Petkov (The Netherlands)
Duc Nghia Pham (Australia)
David Powers (Australia)
Mikhail Prokopenko (Australia)
Arthur Ramer (Australia)
Jochen Renz (Australia)
Anthony Robins (New Zealand)
Panos Rondogiannis (Greece)
Bodo Rosenhahn (Germany)
Malcolm Ryan (Australia)
Rafal Rzepka (Japan)
Ruhul Sarker (Australia)
Torsten Schaub (Germany)
Rolf Schwitter (Australia)
Steven Shapiro (Canada)
Maolin Tang (Australia)
Dacheng Tao (China)
Michael Thielscher (Germany)
Simon Thompson (UK)
Andrea Torsello (Italy)
William Uther (Australia)
Hans van Ditmarsch (New Zealand)
Paolo Viappiani (Canada)
Dianhui Wang (Australia)
Kewen Wang (Australia)
Renata Wassermann (Brazil)
Peter Whigham (New Zealand)
James Whitacre (Australia)
Bill Wilson (Australia)
Brendon Woodford (New Zealand)
Roland Yap (Singapore)
Jian Yu (China)
Lean Yu (China)
Daoqiang Zhang (China)
Dongmo Zhang (Australia)
Min-Ling Zhang (China)
Shichao Zhang (China)
Zili Zhang (Australia)
Yanchang Zhao (Australia)
Yi Zhou (Australia)
Xingquan Zhu (USA)

Additional Reviewers

Vaishak Belle
Loreto Bravo
Nathan Brewer
Duygu Çelik
Weiping Chen
Matthew Damigos
Zafer Erenel
Vladimir Estivill-Castro
Berndt Farwer
Naoki Fukuta
Ana Funes
Masabumi Furuhata
Chrysida Galanaki
Aurona Gerber
Joachim Gudmundsson
Daniel Harabor
Nils Hasler
Saori Kawasaki
Vassilis Kountouriotis
Yat-Chiu Law
Kevin Lee
Wee Sun Lee

Louise Leenen
Richard Leibbrandt
Trent Lewis
Wei Li
Bo Liu
Boon Thau Loo
Christian Moewes
Nobuyuki Morioka
Shinobu Nagayama
Tesuya Nakatoh
Nina Narodytska
Thilky Perera
Laurette Pretorius
Jakob Puchinger
María José Ramírez-Quintana
Wei Ren
Georg Ruß
Scott Sanner
Christian Schmaltz
Andreas Schutt
Matthias Steinbrecher
Xiaoyang Tan
Shanshan Wu

Table of Contents

Invited Paper

Knowledge Representation

Constraints

Planning

Grammar and Language Processing

Statistical Learning

Machine Learning

Data Mining

Knowledge Discovery

Soft Computing

Vision and Image Processing

AI Applications

Stereo-Vision-Support for Intelligent Vehicles - The Need for Quantified Evidence

Reinhard Klette

The *.enpeda..* Project, The University of Auckland
Auckland, New Zealand

Abstract. Vision-based driver assistance in modern cars has to perform automated real-time understanding or modeling of traffic environments based on multiple sensor inputs, using 'normal' or specialized (such as night vision) stereo cameras as default input devices. Distance measurement, lane-departure warning, traffic sign recognition, or trajectory calculation are examples of current developments in the field, contributing to the design of intelligent vehicles.

The considered application scenario is as follows: two or more cameras are installed in a vehicle (typically a car, but possibly also a boat, a wheelchair, a forklift, and so forth), and the operation of this vehicle (by a driver) is supported by analyzing in real-time video sequences recorded by those cameras. Possibly, further sensor data (e.g., GPS, radar) are also analyzed in an integrated system.

Performance evaluation is of eminent importance in car production. Crash tests follow international standards, defining exactly conditions under which a test has to take place. Camera technology became recently an integral part of modern cars. In consequence, perfectly specified and standardized tests ('camera crash tests') are needed very soon for the international car industry to identify parameters of stereo or motion analysis, or of further vision-based components.

This paper reports about current performance evaluation activities in the *.enpeda..* project at The University of Auckland. Test data are so far rectified stereo sequences (provided by Daimler A.G., Germany, in 2007), and stereo sequences recorded with a test vehicle on New Zealand's roads.

Keywords: intelligent vehicle, vision-based driver support, stereo analysis, motion analysis, performance analysis, camera crash tests.

1 Introduction

Current research in vision-based driver assistance asks for the generation of 'ground truth'[1] for real-world sequences, and its use for performance evaluation of various algorithms for stereo image sequence analysis.

[1] The term *ground truth* was coined in photogrammetry when comparing analysis results, derived from aerial imaging, against measured data ('on the ground'). The presence of a measurement error means that ground truth is not truth, but expected to be close to it.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 1–17, 2008.

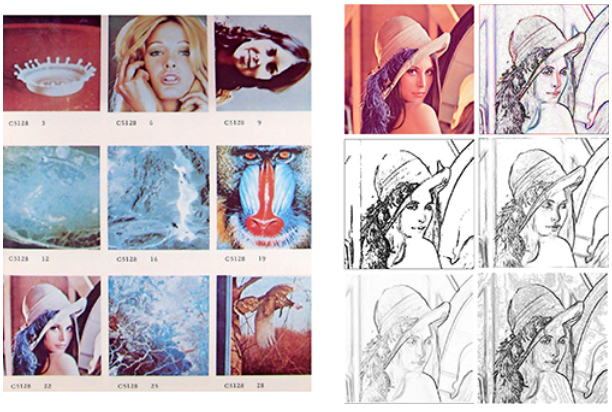


Fig. 1. Left: page in the 1976 report [21], offering eight color images and one multispectral image. Right: 'Lena' and results of various edge detectors. Those 1976 test images are still in use today when demonstrating research on low-level image processing.

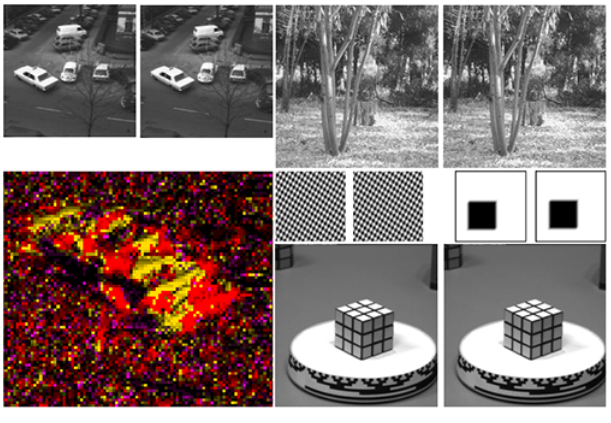


Fig. 2. Rubik cube on a microwave turntable at DEC, the 1971 "Hamburg Taxi", "SRI Trees", and two more "sequences" as used in the 1990*s*. Lower left: color representation of calculated optical flow for the taxi scene. These short sequences did not come with ground truth, and are still used sometimes today (e.g., for student assignments).

Evaluations have a long history in image processing. In a first generation of test images in the 1970*s* (e.g., see [21] for images such as *Lena*, *Mandrill*, *peppers*, *tiffany*, or *zelda*; "copies of the IPI data base" were "supplied on magnetic tape, 9 track, 800 BPI, on 2400-ft. reels"; Fig. 1 shows nine of those test images), there were no stereo images, and no image sequences at all at that time in the test data base. Very short sequences of images became popular in the 1980*s*, such as those shown in Fig. 2, which allowed to compare results for optical flow. The lower left in Fig. 2 shows a calculated vector field (as obtained in a student assignment in the 1990*s* in one of my classes) in common hue-intensity

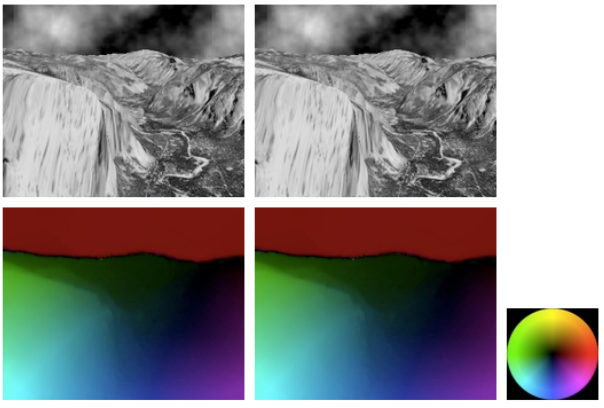


Fig. 3. Demonstration of calculated optical flow [23], using the 1984 Yosemite sequence as discussed on [2]. This sequence is still a popular way for demonstrating optical flow results.

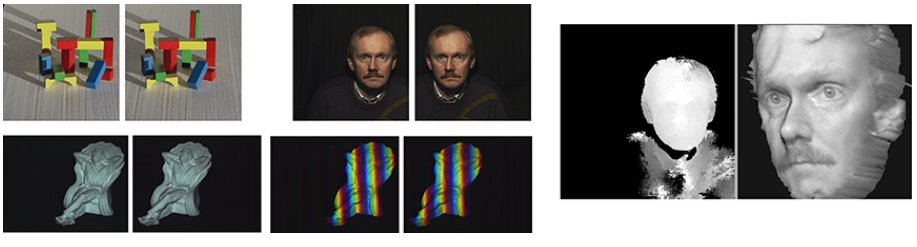


Fig. 4. Stereo pairs as used in the 1996 textbook [11] for evaluating the performance of various stereo matching algorithms. An example of a reconstructed face is shown on the right.

representation. The taxi sequence was actually recorded in 1971 (!) in the group of H.-H. Nagel [17].

The Yosemite sequence (by L. Quam [19]; see Fig. 3) "has been used extensively for experimentation and quantitative evaluation of optical flow methods, camera motion estimation, and structure from motion algorithms." [2] This is a synthetic sequence of 316×252 images, simulating a flight through a valley, with ground truth motion data (quantized to 8 bits) for rigid objects in the scene.

Test data for stereo analysis should be in standard binocular stereo geometry; [11] offered those based on using an optic bench in the lab and careful camera adjustments; see Fig. 4. There was no ground truth provided, and evaluation was based on subjective (visual) comparisons.

Automated stereo pair rectification [15] maps today stereo images into standard binocular stereo geometry [8]. This allows to generate sets of stereo images, ready for correspondence analysis. Laser-range finders may be used to generate ground truth for such stereo images by modeling real scenes [9].

Fig. 5. Illustration of four stereo image sets on the Middlebury vision website: *map*, *sawtooth*, *venus*, and *Tsukuba* with depth map, illustrating ground truth as available for those data sets on this website

Fig. 6. An illustration for seven stereo night vision sequences available since 2007 on the *.enpeda..* website [4] for performance evaluation. Left: one out of close to 2,000 rectified stereo pairs in total. Right: screenshot of an avi showing one original sequence (lower left) and disparity data.

The Middlebury vision page [16] (of D. Scharstein, R. Szeliski, et al.) provided in its 'pre-2008-web-server-crash' version only a few engineered samples of input images for stereo and motion analysis; see four stereo sets illustrated in Fig. 5. This website stimulates current progress in computer vision (and a web-server crash in August 2008 was followed with eagerness in the computer vision community worldwide). Currently the website is revised, now also featuring more data sets for performance evaluation, but still focussing on indoor, engineered, high contrast imagery.

Driver assistance systems (DAS), see, for example, [5], the monograph [3] of E.D. Dickmanns, or proceedings [22], have to deal with stereo image sequences

Fig. 7. Test vehicle HAKA1 with a pair of cameras for stereo image sequence capture, recording stereo sequences on Auckland's roads since July 2008

recorded under any possible weather or lighting condition. See Fig. 6 for an illustration of DAS stereo sequences: seven rectified night-vision stereo sequences are available since 2007 on the *.enpeda..* website [4] for motion and stereo performance evaluation; the sequence data have been provided by Daimler AG (group of U. Franke) and prepared in 2007 by T. Vaudrey and Z. Liu for online presentation (with camera calibration and motion data for the ego-vehicle).

DAS sequences may contain unpredictable events and all kinds of variations in recorded image data, for example due to a partially 'faulty' camera, generating more blurry images in the left camera than in the right camera, or due to different brightness in left and right camera. More rectified stereo real-world sequences will be made available on the *.enpeda..* website [4] soon, including those recorded with a test vehicle (HAKA1, 'High Awareness Kinematic Automobile no. 1') in Auckland (see Fig. 7).

Obviously, it is a challenge to provide ground truth (3D environment, poses of agents) for such sequences. Three approaches appear to be possible options for satisfying the needs of *camera crash tests* as indicated in the Abstract of this paper:

(1) Post-modeling of recorded 3D environments: based on recorded stereo sequences, apply (possibly manual) 3D modeling software to generate a 3D dynamic model of the recorded scene.
(2) Accumulated evidence for 3D environments: in extension of the post-modeling approach, drive repeatedly into the same (static) 3D environment, and attempt to improve the 3D model (shape plus texture) by accumulation, merging, or unification of obtained 3D data (also using other sensors).
(3) Pre-modeling of recorded 3D environments: use 3D modeling approaches such as laser-range finders or sensor technology to generate an accurate 3D model (shape plus texture) of a defined environment and operating *agents* (vehicles or persons), and of poses of ego-vehicle and also of agents during recording.

Fig. 8. Examples of manually specified rectangular regions for approximated ground truth: in original sequences of Set 1 on [4] (left) and in Sobel-based BP results (right)

This paper will report in the second section about work towards the first approach. For the second or third approach, see, for example, [6], where also a laser-range finder is mounted on a mobile platform, used for modeling city scenes. Laser-range finders allow very accurate 3D large-scale models, see [9]. For example, a particular area might be 3D modeled, such as a courtyard which is basically 'static', and this area may then serve as a 'camera crash test site', similar to crash test halls at car companies. For combining various sensors for

3D modeling, see, for example, [12]. Alternatively, large scale modeling may also utilize technology as developed for the generation of 3D maps [1], also discussed in [9].

2 Approximate Ground Truth

In a recorded stereo sequence, we may identify simple geometric shapes and identify their 3D location, using automated or manual measurements; see Fig. 8. (The figure also shows identified rectangular areas in depth maps calculated using belief propagation as specified in [7].) As a more general option [14], we may assume an approximate planar road surface, using known parameters of ego-vehicle and cameras (as saved for Set 1 on [4] in the *camera.dat* file and in the file header of every frame; see [13]).

2.1 Disparities on Road Surface

We assume that test sequences are ego-motion compensated, which means that the horizon is always parallel with the row direction in the images, and pixels on the same image row have the same depth value if a projection of the planar road surface.

A side-view of the camera setting is shown in Figure 9, where θ is the tilt angle, P is a road surface point which is projected into $p = (x_p, y_p)$ on the image plane, H is the height of the camera. It follows that

$$Z = d_e(OP_c) = d_e(OP)\cos\psi = \frac{H}{\sin(\theta+\psi)}\cos\psi \tag{1}$$

According to the stereo projection equations, the disparity d can be written as

$$d = \frac{b \cdot f}{Z} = \frac{b \cdot f}{\frac{H}{\sin(\theta+\psi)}\cos\psi} \tag{2}$$

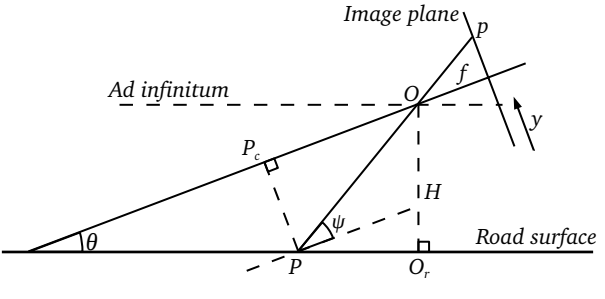


Fig. 9. Projection of a point P of the road surface

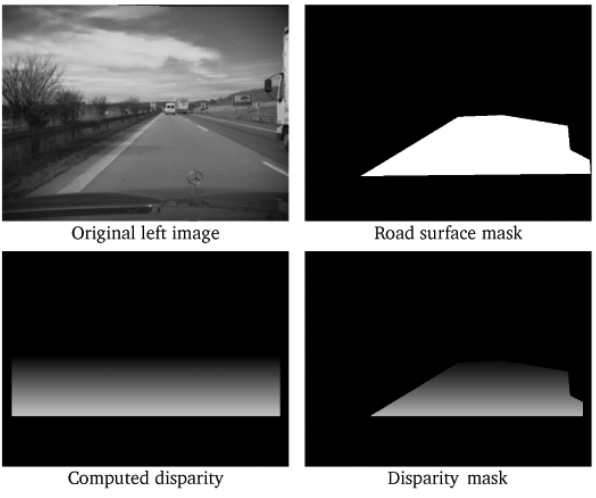


Fig. 10. Generation of a disparity mask: input image, generated road mask, depth map of a planar road, and resulting disparity mask

where angle ψ can be calculated as follows, using focal length f and pixel coordinate y_p in the image:

$$\psi = \arctan\left(\frac{(y_p - y_0)s_y}{f}\right) \tag{3}$$

Here, y_0 is the y-coordinate of the principal point, and s_y is the pixel size in y-direction. We can also compute the y-coordinate of a line that projects to infinity

$$y_{inf} = \frac{y_0 - f \cdot \tan\theta}{s_y}$$

This is the upper limit of the road surface, and points on it should have zero disparity (if no objects block the view).

Figure 10 illustrates the process of generating an approximated disparity map on road surface areas, also using manual input for a conservative outline of the road area in a given image. In the given camera setting (of the seven sequences), there is a yaw angle (0.01 radian) which makes the cameras looking a little bit to the left. This angle can be ignored because it only defines the right camera to be about 3 mm behind the left camera.

2.2 Recalibration of Tilt Angle

Although a camera tilt angle is already given for these sequences, we noticed that the angle is not always true when verifying the data. This problem might be caused by several reasons, for example, the road surface is changing (downhill, uphill), the car coordinate system is not parallel to the road surface in some

Table 1. Results of tilt angle estimation for the given seven sequences

Sequence name	Tilt angle (radian)
1: 2007-03-06_121807	0.01608
2: 2007-03-07_144703	0.01312
3: 2007-03-15_182043	0.02050
4: 2007-04-20_083101	0.06126
5: 2007-04-27_145842	0.06223
6: 2007-04-27_155554	0.06944
7: 2007-05-08_132636	0.05961

situations (acceleration, braking), drivers of different weight, or driving with flat tires, or the installation of cameras may change for some reasons. (Actually, changes are easy to detect by reading the position of the Mercedes star in the given images.)

The outlined process for obtaining approximate stereo ground truth identified the importance of the tilt angle for the estimated values. We propose a method to estimate the average tilt angle for a given sequence of frames. This method is similar to the road surface stereo approximation, just in a reverse order. We estimate the tilt angle based on given depth at some feature points (i.e., with known disparities) which can be measured or identified manually.

See Figure 10 and assume a given pair of corresponding points, with disparity d. By Equation (2) we have that the tilt angle can be written as follows:

$$\theta = \arcsin\big(\frac{H\cos\psi \cdot d}{b \cdot f}\big) - \psi \tag{4}$$

where ψ is as given in Equation (3).

Altogether, at first, we randomly select five or six frames from a sequence of frames, then, we calculate or choose pairs of corresponding pixels on the road surface area, and obtain disparities between those. Each disparity (of one pixel pair) can be used to calculate a tilt angle using Equation (4), and a mean of those provides a tilt angle estimation; see Table 1 for results for the seven sequences.

2.3 2D Motion on Road Surface

Speed and direction (yaw rate) of the ego-vehicle are given for all frames of those seven sequences. The road is, obviously, static, what makes the calculation of relative movement of road surface points (with respect to the camera) straight forward.

Given a pixel p on the image plane at time t, which is projected to a road surface point P. Let P move to a new position P' at time $t + \delta t$, where δt is the time interval between two consecutive frames (called `CycleTime` in the seven sequences, either equals 0.04 s or 0.08 s). Then, P' is projected back to the image plane at p'; see Figure 11. The approximation of 2D motion (i.e., local displacement) at a pixel can then proceed as follows:

First, assume that the vehicle speed equals $\mathbf{v}$ at time t, and $\mathbf{v}'$ at time $t + \delta t$; the average speed during this time interval equals δt is $\overline{\mathbf{v}} = \frac{\mathbf{v}+\mathbf{v}'}{2}$, having δt very

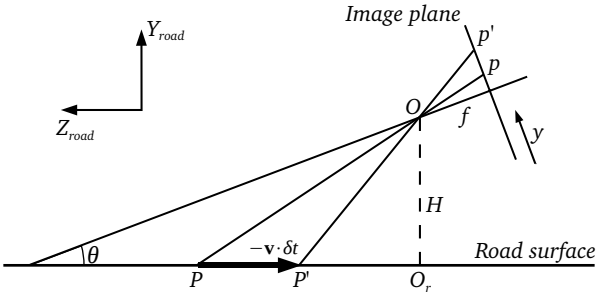


Fig. 11. Approximation of 2D motion in y-direction: P and P is the same road surface point, just in two consecutive frames. P is projected into $p = (x, y)$ in the image plane, P is projected into $p = (x , y)$.

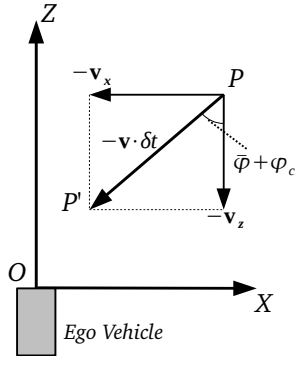


Fig. 12. Change in relative position between road surface point P and ego-vehicle

small in the sequences. Distances (in Z_{road} coordinates) of moving points are defined as follows:

$$d_Z(P, P') = |\overline{\mathbf{v}}| \cos(\overline{\varphi} + \varphi_c)\delta t = \frac{|\mathbf{v}_1| + |\mathbf{v}_2|}{2} \cos(\frac{\varphi_1 + \varphi_2}{2} + \varphi_c)\delta t$$

where φ_1 and φ_2 are the yaw angles of the ego-vehicle at t and $t + 1$, and φ_c is the yaw angle of the camera installation (see Figure 12). Therefore, the distance between the point P and the ego-vehicle becomes

$$Z_{P'} = d_Z(O_r, P') = d_Z(O_r, P) - d_Z(PP') = \frac{H}{\tan(\theta + \psi)} - d_Z(PP')$$

Then, the angle between the projection ray OP' and the optical axis of the camera may be determined as follows:

$$\psi' = \arctan\big(\frac{H}{d_Z(O_r, P')}\big) - \theta = \arctan\big(\frac{H}{d_Z(O_r, P) - d_Z(P, P')}\big) - \theta$$

where $d_Z(O_r, P) = \frac{H}{\tan(\theta+\psi)}$.

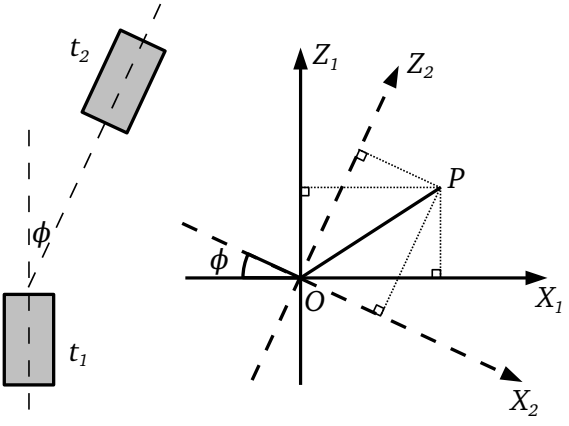


Fig. 13. A rotation of the ego-vehicle

Therefore, according to Equation (3), the y-coordinate of 2D motion $\mathbf{u}$ at point P' can be written as

$$v = \big(\frac{f \cdot \tan(\psi')}{s_y} + y_0\big) - y_p$$

Thus, we are also able to specify the position of point P in x-direction as follows

$$X_P = \frac{Z_P \cdot x_p}{f}$$

with $Z_P = \frac{H}{\sin(\theta+\psi)} \cos\psi$, which is actually already a known value from the previous stereo ground truth approximation.

The position of P' (for the next frame) can then be calculated by using speed $\mathbf{v}$ and time interval δt,

$$X_{P'} = X_P - |\mathbf{v}| \sin(\overline{\varphi} + \varphi_c)\delta t$$

Now we have the new relative position between the road surface point and the vehicle at time $t + \delta t$. - In a next step, we need to rotate the vehicle coordinate system by an angle according to the yaw rate given in the vehicle movement parameters; see Figure 13. Therefore, the final (relative) position equals

$$\begin{bmatrix} X^{\phi}_{P'} \\ Z^{\phi}_{P'} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} X_{P'} \\ Z_{P'} \end{bmatrix}$$

In a final step, point P is projected back to a pixel p' on the camera's image plane. Then, 2D motion is obtained by comparing locations of p and p', as follows:

$$\psi' = \arctan\big(\frac{H}{Z^{\phi}_{P'}}\big) - \theta$$

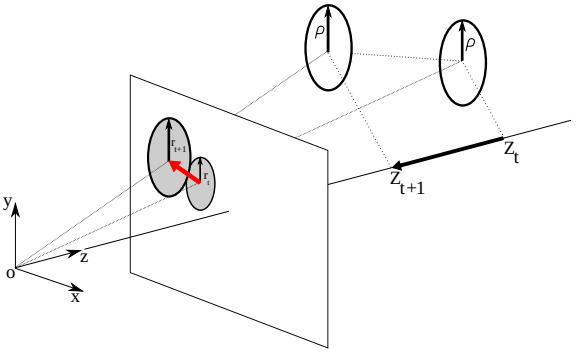


Fig. 14. Two projections of a moving disk, at times t and $t+1$

$$v = y'_p - y_p = \left(\frac{f \cdot \tan(\psi')}{s_y} + y_0\right) - y_p$$

$$u = x_{p'} - x_p = \frac{f \cdot X^{\phi}_{P'}}{\frac{H}{\sin(\theta+\psi')}\cos(\psi')} - x_p$$

2.4 Change in Depth for Image Features

As another option for modeling recorded scenes, we may use a scale-space based estimation of changes in depth [20]. Consider a disk of radius ρ moving towards an ideal pinhole-type camera of focal length f. Without loss of generality, let the radius move parallel to the Y-axis of the XYZ-camera coordinate system (i.e., $r = Y_c - Y_e$, for center P_c and an edge point P_e of the disk). A 3D point $P = (X, Y, Z)$ in the world (in camera coordinates) projects into a point $p = (x, y, f)$ in the image plane, with $x = f\frac{X}{Z}$ and $y = f\frac{Y}{Z}$. Point P_c projects into $p_c = (x_c, y_c, f)$, and P_e projects into $p_e = (x_e, y_e, f)$. The moving disk is at time t at distance Z_t, and projected into image I_t as a disk of radius r_t (see Fig. 14). We obtain the following for the area of this projected disk:

$$A_t = \pi r_t^2 = \pi \left(y_c - y_e\right)^2 = f\frac{\pi}{Z_t^2}\left(Y_c - Y_e\right)^2 = \pi f \frac{\rho^2}{Z^2}$$

Radius ρ of the disk is constant over time, thus, the product $A_t Z_t^2 \sim \rho^2$ will also not change over time.

We consider projections of the disk at times t and $t+1$. Because the ratio of square roots of areas is proportional to the inverse of the ratio of corresponding Z-coordinates of the disk, we are able to define a *z-ratio*

$$\mu_z = \frac{\sqrt{A_t}}{\sqrt{A_{t+1}}} = \frac{Z_{t+1}}{Z_t} \tag{5}$$

either by area or Z-values.

Such a z-ratio can also be defined just for a pair of projected points $P_t = (X_t, Y_t, X_t)$ and $P_{t+1} = (X_{t+1}, Y_{t+1}, Z_{t+1})$ (just by the ratio of Z-coordinates). Using the central projection equations for both projected points, we obtain for their *x-ratio* and *y-ratio* the following:

$$\mu_x = \frac{X_{t+1}}{X_t} = \frac{Z_{t+1}}{Z_t} \cdot \frac{x_{t+1}}{x_t} = \mu_z \frac{x_{t+1}}{x_t} \tag{6}$$

$$\mu_y = \frac{Y_{t+1}}{Y_t} = \frac{Z_{t+1}}{Z_t} \cdot \frac{y_{t+1}}{y_t} = \mu_z \frac{y_{t+1}}{y_t} \tag{7}$$

Altogether, this may also be expressed by the following *update equation*:

$$\begin{pmatrix} X_{t+1} \\ Y_{t+1} \\ Z_{t+1} \end{pmatrix} = \begin{pmatrix} \mu_x & 0 & 0 \\ 0 & \mu_y & 0 \\ 0 & 0 & \mu_z \end{pmatrix} \begin{pmatrix} X_t \\ Y_t \\ Z_t \end{pmatrix} \tag{8}$$

with μ_x, μ_y, and μ_z as in Equations (6), (7), and (5) respectively. In other words, knowing μ_z and ratios $\frac{x_{t+1}}{x_t}$ and $\frac{y_{t+1}}{y_t}$ allows to update the position of point P_t into P_{t+1}. Assuming that P_t and P_{t+1} are positions of one tracked 3D point P, from time t to time $t+1$, we only have to solve two tasks: (1) decide for a technique to track points from t to $t+1$, and (2) estimate μ_z. If an initial position P_0 of a tracked point P is known then we may identify its 3D position at subsequent time slots. Without having an initial position, we only have a 3D direction P_t to P_{t+1}, but not its 3D position.

For identifying μ_z, an 'area of influence' is assigned to each tracked feature point, basically taking the role of a tracked disk.

For tracked points, a scale-space-based measure is computed for the 'extension of the local image structure' in a local (or semi-local) neighborhood. Such

Fig. 15. Disks with radii defined by maxima of scale space characteristics

measures, computed independently for each pair of points, are used to determine a scale ratio (based on associated intensity profiles of scale characteristics of those feature points), which is finally used as an estimate of the z-ratio μ_z. For details, see [20]. Figure 15 illustrates disks assigned to tracked features.

3 Evaluation

We use quality metrics to measure the quality of calculated stereo correspondences or motion vectors with respect to approximated ground truth.

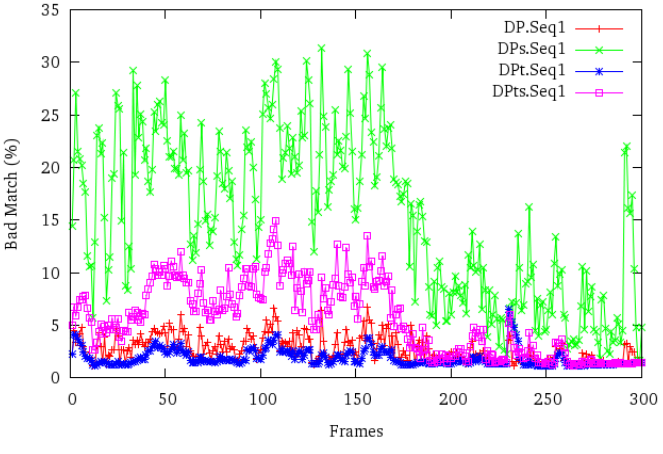


Fig. 16. Percentages of bad matches for dynamic programming stereo and its variants

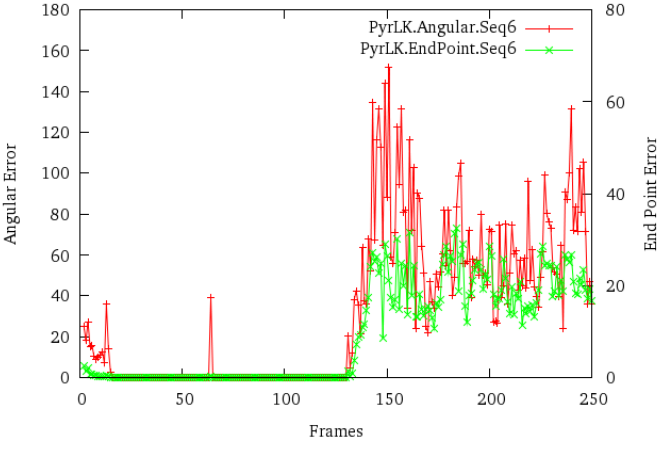


Fig. 17. Angular errors and endpoint errors for PyrLK on Sequence 6

3.1 Stereo

The general approach of stereo evaluation is to compute error statistics based on given ground truth. We use the same error measurements as on [16], namely the *root mean squared error* between the disparity map $d(x, y)$ and the ground truth map $d_T(x, y)$, defined as follows:

$$E_R = (\frac{1}{n} \sum |d(x, y) - d_T(x, y)|^2)^{\frac{1}{2}} \tag{9}$$

where n is the total number of pixels, and the percentage of *bad matching pixels*, defined as follows:

$$E_B = \frac{1}{n} \sum (|d(x, y) - d_T(x, y)| > \delta_d) \tag{10}$$

where δ_d is the threshold of disparity tolerance.

Quality metrics for optical flow evaluation have to measure the result in a 2D space. We use the common *angular error* defined as the average angle between estimated optical flow vector $\mathbf{u}$ and the true flow vector $\mathbf{u}_T$,

$$E_{AE} = \frac{1}{n} \sum \arccos\left(\frac{\mathbf{u} \cdot \mathbf{u}_T}{|\mathbf{u}||\mathbf{u}_T|}\right) \tag{11}$$

where $|\mathbf{u}|$ denotes the length (magnitude) of a vector, and the *end point error* which measures the absolute distance between the end points of vectors $\mathbf{u}$ and $\mathbf{u}_T$,

$$E_{EP} = \sqrt{(u - u_T)^2 + (v - v_T)^2} \tag{12}$$

3.2 Examples of Results

The discussed approximate ground truth has been used in [7,14] for evaluating stereo and motion analysis techniques, such as variants of dynamic programming (including Birchfield-Tomasi), belief propagation, semi-global matching, or variants of optical flow calculation (using sources in OpenCV [18] where available, D. Huttenlocher's belief propagation sources from [10], or our own implementation).

For example, Fig. 16 shows bad matches for Sequence 1 (of Set 1 on [4]), comparing a common dynamic programming approach with modifications, also using spatial or temporal propagation (only one of those, or both combined). The figure shows values for all the 300 stereo pairs of this sequence. It clearly indicates that temporal propagation (see DPt in the diagram) is of benefit if evaluating within the described road mask of estimated disparities.

Figure 17 summarizes angular and end point errors of the pyramid Lucas-Kanade technique for all 250 frames of the left camera of Sequence 6.

We will not start a comparative discussion here, and point the reader to [7,14]. The two examples of diagrams are given here to illustrate an important property of these evaluations based on real-world sequences: here we have long sequences, basically of arbitrary length, and we may use this for improving results (e.g., by applying a Kalman filter), but also for deriving statistically more relevant performance evaluation results.

4 Conclusions

Vision-based driver assistance systems have moved into modern cars in recent years, and there will be an 'exponential growth' in demands not only with respect to deriving accurate and real-time computer vision solutions, but also in evaluating these solutions, to ensure that they satisfy international standards (still to be defined by major car manufacturers).

This will require that testing is based on real-world data, without eliminating any possible visual effect, and with aiming at 'robust' testing. A vision system may be 'robust' if being fairly invariant with respect to changes in brightness or contrast; obviously, a smoke detection system should not have this type of 'robustness'. We conclude that 'robustness' needs to be defined for the particular needs of DAS.

Evaluation not only needs to be done *also* on stereo real-world sequences; we may expect that the car industry will define the state of the art in stereo and motion analysis with their (expected) quality standards very soon. Image analysis will also work on rainy days, even in the night, and so forth.

Acknowledgement. The author acknowledges valuable support of, or collaboration with (in alphabetic order) Je Ahn, Ali Al-Sarraf, Eduardo Destefanis, Shushi Guan, Zhifeng Liu, Jorge Sánchez, and Tobi Vaudrey.

References

1. 3D Reality Maps^{TM}, `http://www.realitymaps.de/`
2. Black, M.: Comments about the Yosemite sequence, `http://www.cs.brown.edu/ black/Sequences/yosFAQ.html`
3. Dickmanns, E.D.: Dynamic Vision for Perception and Control of Motion. Springer, London (2007)
4. Enpeda. Image sequence analysis test site, `http://www.mi.auckland.ac.nz/`
5. Franke, U., Gavrila, D., Gorzig, S., Lindner, F., Paetzold, F., Wöhler, C.: Autonomous driving goes downtown. IEEE Int. Systems 13, 40–48 (1998)
6. Früh, C., Zakhor, A.: An automated method for large-scale, ground-based city model acquisition. Int. J. Computer Vision 60, 5–24 (2004)
7. Guan, S., Klette, R.: Belief propagation for stereo analysis of night-vision sequences. Technical report, Computer Science Department, The University of Auckland (2008)
8. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
9. Huang, F., Klette, R., Scheibe, K.: Panoramic Imaging - Rotating Sensor-Line Cameras and Laser Range-Finders. Wiley, Chichester (2008)
10. Huttenlocher, D.: Loopy belief propagation sources, `http://people.cs.uchicago.edu/ pff/bp/`
11. Klette, R., Koschan, A., Schlüns, K.: Computer Vision. Vieweg, Braunschweig (1996)
12. Klette, R., Reulke, R.: Modeling 3D scenes: paradigm shifts in photogrammetry, remote sensing and computer vision. Opening Keynote. In: Proc. Int. IEEE Conf. ICSS, Taiwan (2005) (on CD, 8 pages), `http://www.citr.auckland.ac.nz/techreports/show.php?id=155`

13. Liu, Z., Klette, R.: Performance evaluation of stereo and motion analysis on rectified image sequences. Technical report, Computer Science Department, The University of Auckland (2007)
14. Liu, Z., Klette, R.: Approximated ground truth for stereo and motion analysis on real-world sequences. Technical report, Computer Science Department, The University of Auckland (2008)
15. Longuet-Higgins, H.C.: A computer algorithm for reconstructing a scene from two projections. Nature 293, 133–135 (1981)
16. Middlebury vision website, http://vision.middlebury.edu/
17. Nagel, H.-H.: Image sequence evaluation: 30 years and still going strong. In: Proc. ICPR, vol. 1, pp. 149–158 (2000)
18. Open Source Computer Vision Library, http://www.intel.com/research/mrl/research/opencv/
19. Quam, L.: Hierarchical warp stereo. In: Proc. DARPA Image Understanding Workshop, pp. 149–155 (1984)
20. Sánchez, J., Klette, R., Destefanis, E.: Estimating 3D flow for driver assistance applications. Technical report, Computer Science Department, The University of Auckland (2008)
21. Schmidt, R.: The USC - Image Processing Institute data base, revision 1. USCIPI Report 780 (October 1976)
22. Sommer, G., Klette, R. (eds.): RobVis 2008. LNCS, vol. 4931. Springer, Heidelberg (2008)
23. Weickert, J., et al.: Online presentation of MIA Group, http://www.mia.uni-saarland.de/OpticFlow.shtml

Introspective Forgetting

Hans van Ditmarsch[1,2,⋆], Andreas Herzig[2], Jérôme Lang[2], and Pierre Marquis[3]

[1] Computer Science, University of Otago, New Zealand
hans@cs.otago.ac.nz
[2] IRIT, Université Paul Sabatier, France
{hans,herzig,lang}@irit.fr
[3] CRIL, Université d'Artois, France
marquis@cril.univ-artois.fr

Abstract. We model the forgetting of propositional variables in a modal logical context where agents become ignorant and are aware of each others' or their own resulting ignorance. The resulting logic is sound and complete. It can be compared to variable-forgetting as abstraction from information, wherein agents become unaware of certain variables: by employing elementary results for bisimulation, it follows that beliefs not involving the forgotten atom(s) remain true.

Keywords: modal logic, forgetting, abstraction, action logic, belief change.

1 There Are Different Ways of Forgetting

Becoming unaware. In the movie 'Men in Black', Will Smith makes you forget knowledge of extraterrestrials by flashing you with a light in the face. After that, you have forgotten the green ooze flowing out of mock-humans and such: you do not remember that you previously had these experiences. In other words, even though for some specific forgotten fact p it is now the case that $\neg Kp$ and $\neg K\neg p$, the flash victims have no memory that they previously knew the value of p. Worse, they forgot that p is an atomic proposition at all. This sort of forgetting is dual to awareness—in a logical setting this means that parameters of the language, such as the set of atoms, shrink.

Becoming ignorant. A different sort of forgetting is when you forgot which of two keys fits your office door, because you have been away from town for a while. Is it the bigger or the smaller key? This is about forgetting the value of an atomic proposition p—such as "the bigger key fits the door." You are embarrassingly aware of your current ignorance: introspection is involved. We have $K(\neg Kp \wedge \neg K\neg p)$. This sort of forgetting is central to our concerns.

Remembering prior knowledge. You also remember that you *knew* which key it was. You just forgot. Previously Kp or $K\neg p$, and only now $\neg Kp$ and $\neg K\neg p$.

⋆ Corresponding author.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 18–29, 2008.

Forgetting values. Did it ever happen to you that you met a person whose face you recognize but whose name you no longer remember? Surely! Or that you no longer know the pincode of your bankcard? Hopefully not. But such a thing is very conceivable. This sort of forgetting means that you forgot the value of a proposition, or the assignment of two values from different sets of objects to each other. An atomic proposition about your office keys is a feature with two values only, true and false. The (finitely) multiple-valued feature can be modelled as a *number* of atomic propositions. Forgetting of such multiple boolean variables is in our approach similar to forgetting a single boolean variable.

Multi-agent versions of forgetting. Will Smith only had to flash a whole group once, not each of its members individually. So, in a multi-agent setting some aspects of collectively 'becoming unaware' can be modelled. A different, familiar phenomenon is that of an individual becoming unaware in a group: "You forgot my birthday, *again*!"

A group version for 'remembering prior knowledge' would involve common awareness, and prior common knowledge. This collective introspective character is not always easy to justify. On the other hand, a version of 'remembering prior knowledge' for *individuals* in a group is more intuitive, because they can inform and are observed by others: here you are standing in front of your office door again, now in company of four freshmen students, "Ohmigod, I forgot again which is my office key!"

I may have forgotten whether you knew about a specific review result for our jointly editored journal issue. In other words, previously $K_{me}K_{you}\,accept$ or $K_{me}K_{you}\neg accept$ but currently $\neg K_{me}K_{you}\,accept$ and $\neg K_{me}K_{you}\neg accept$. Some meaningful propositions that can be forgotten in a multi-modal context are therefore themselves modal.

Forgetting events. Say I forgot to pick you up at the airport at 4:30 PM. Forgetting an *action* (event) is different from forgetting a *proposition*. 'Forgetting of events' amounts to introducing *temporal uncertainty* in the model, apart from epistemic uncertainty. The *observation* of having forgotten it, is about the recovery that takes place after forgetting the event.

2 Motivation

A short history of forgetting in AI. In 'Forget it!' [1] Lin and Reiter proposed a way to abstract from ground atoms (that can be equated to propositional variables) in a set of first-order beliefs, employing the notion of *similarity of models for a theory except for such a ground atom*. They leave it open whether such forgetting is the result of an agent consciously updating a knowledge base after having learnt about *factual* change, or whether this is simple erosion of her working memory, purely *epistemic* change. Their work was built upon by Lang, Liberatore and Marquis with their in-depth study on the computational costs of transforming theories by variable forgetting[2], or rather the costs of determining the independence of parts of a theory from specific variables. In [3]

Baral and Zhang took part of this battleground to involve more explicit operators for knowledge and belief, where the result of an agent forgetting a variable results in her (explicit) ignorance of that variable's value, and in [4], in progress, Zhang and Zhou make an original and interesting backtrack to the ideas of [1] by suggesting *bisimulation invariance except for the forgotten variable*, in order to model forgetting. Forgetting has been generalized to logic programs in [5,6,7] and to description logics in [8]. Forgetting of (abstracting from) actions in planning has been investigated in [9].

Progression of belief sets. Generalizing the results in [2] to forgetting in *positive* epistemic formulas (subformulas expressing ignorance are not allowed) is easy [10], but beyond that it is hard. Consider the binary operation

$$Fg(\Phi, p) \equiv_{\mathsf{def}} \{\varphi(\top/p) \vee \varphi(\bot/p) \mid \varphi \in \Phi\}$$

wherein $\varphi(\psi/p)$ is the replacement of all (possibly zero) occurrences of p in φ by ψ. This defines the (syntactic) *progression* of Φ when forgetting about p. When applying this recipe to formulas expressing ignorance, we get undesirable consequences, e.g. that $Fg(\neg Kp \wedge \neg K\neg p, p)$ is equivalent to the contradiction $\bot$. This is shown as follows:

$Fg(\neg Kp \wedge \neg K\neg p, p)$
is by definition
$(\neg K\top \wedge \neg K\neg\top) \vee (\neg K\bot \wedge \neg K\neg\bot)$
iff
$(\neg\top \wedge \neg\bot) \vee (\neg\bot \wedge \neg\top)$
iff
$\bot$

Such problems motivated us to model forgetting as an event in a dynamic epistemic logic.

Forgetting as a dynamic modal operator. We model the action of forgetting an atomic proposition p as an *event* $Fg(p)$. We do this in a propositional logic expanded with an epistemic modal operator K and a dynamic modal operator $[Fg(p)]$, with obvious multiple-value and multi-agent versions. Formula $[Fg(p)]\varphi$ means that after the agent forgets his knowledge about p, φ is true. We call $[Fg(p)]$ a *dynamic* modal operator because it is interpreted by a state transformation, more particularly: by changing an information state that is represented by a pointed Kripke model (M, s) into another information state (M', s'). The relation to the theory transforming operation $Fg(\Phi, p)$ is as follows: for all models (M, s) of Φ, $[Fg(p)]\varphi$ should be true in (M, s) if and only if $\varphi \in Fg(\Phi, p)$.

A *precondition* for event $Fg(p)$ seems prior knowledge of the value of p: $Kp \vee K\neg p$. How can you forget something unless you know it in the first place? To make our approach comparable to variable forgetting in the 'abstracting-from-information'-sense, we do not require prior knowledge as a precondition for forgetting. The obvious *postcondition* for event $Fg(p)$ is ignorance of the value of p: $\neg Kp \wedge \neg K\neg p$. It should therefore be valid that

$$[Fg(p)](\neg Kp \wedge \neg K\neg p).$$

Forgetting or no-forgetting? On ontic and epistemic change. Wasn't dynamic epistemic logic supposed to satisfy the principle of 'no forgetting' (a.k.a. 'perfect recall')? This entails that positive knowledge such as factual knowledge Kp and $K\neg p$, is preserved after any event. Or, dually: if you are ignorant about p now, then you must have been ignorant about p before. So how on earth can one model forgetting in this setting? We can, because we cheat. We solve this dilemma by the standard everyday solution of forgetful people: blame others. In this case: blame the world; we *simulate* forgetting by *changing the value of p in the actual or other states, in a way known to be unobservable by the agent.* Thus resulting in her ignorance about p.[1]

Having cheated in that way, our logic is equivalent to one without actual change of facts in the one and only way that counts: it makes no difference for believed formulas, i.e., for expressions of the form $K\varphi$.

Remembering prior knowledge. To express that an agent recalls prior knowledge we have to be able to refer to past events. Let $Fg(p)^-$ be the converse of $Fg(p)$ (e.g. in the sense of [13,14,15]). We can now express prior knowledge of now forgotten variables as

$$K(\neg Kp \wedge \neg K\neg p \wedge \langle Fg(p)^-\rangle(Kp \vee K\neg p))$$

This stands for: "the agent knows that (she does not know p and she does not know $\neg p$ and before forgetting about p she either knew p or knew $\neg p$). We will outline our progress towards modelling this in the concluding section.

3 A Logic of Propositional Variable Forgetting

We present a single agent and single variable version of the logic *only.* All results trivially generalize to multiple agents and multiple values (see page 26).

Language, structures and semantics. Given is a set P of propositional variables.

Definition 1 (Language and structures). *Our language $\mathcal{L}$ is*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K\varphi \mid [Fg(p)]\varphi$$

where $p \in P$, and our structures are pointed Kripke models $((S, R, V), s)$*, with $R \subseteq (S \times S)$, $V : P \to \mathcal{P}(S)$, and $s \in S$.*

[1] This is different from how belief revision is modelled in dynamic epistemic (doxastic) logic. Prior belief in p that is revised with $\neg p$ and results in belief in $\neg p$ is standardly modelled by considering this a 'soft' or defeasible form of belief, i.e., not knowledge, and implemented by changing a preference relation between states [11,12].

If $P' \subseteq P$, then $\mathcal{L}(P')$ is the language restricted to P'. The diamond versions of our modal operators are defined as $\hat{K}\varphi \equiv_{\mathsf{def}} \neg K \neg \varphi$ and $\langle Fg(p)\rangle\varphi \equiv_{\mathsf{def}} \neg[Fg(p)]\neg\varphi$. The structures are typically $S5$ to model knowledge and $KD45$ to model belief—but this is not a requirement.

The dynamic operator $[Fg(p)]$ is relative to the state transformer $Fg(p)$ that is an *event model*. The pointed Kripke models are static structures, encoding knowledge and belief, and the event models are dynamic structures, encoding *change of* knowledge and belief. Formally, *multiple-pointed event models* (a.k.a. action models) are structures $(\mathsf{M}, \mathsf{S}') = ((\mathsf{S}, \mathsf{R}, \mathsf{pre}, \mathsf{post}), \mathsf{S}')$, where $\mathsf{S}' \subseteq \mathsf{S}$, where $\mathsf{pre} : \mathsf{S} \to \mathcal{L}$ assigns to each event $\mathsf{s} \in \mathsf{S}$ a *precondition* and where $\mathsf{post} : \mathsf{S} \to (P \to \mathcal{L})$ assigns to each event a *postcondition* (a.k.a. assignment) for each atom (of a *finite* subset of all atoms—the remaining atoms do not change value). For such event models see [16,17]—we follow notational conventions as in [17]. If $\mathsf{post}(\mathsf{s})(p) = \psi$, then we also write that $p := \psi$ (the valuation of atom p becomes that of formula ψ) in the event of s. Dynamic operators expressing event model execution (*semantics*) can be seen as part of the logical language (*syntax*), similar to how this is done for automata-PDL [18].

Forgetting $Fg(p)$ is the event model that expresses that the agent cannot distinguish between two assignments having taken place: p becomes true, or p becomes false. It consists of two events, that are both points (this expresses non-determinism). Both events are always executable: their precondition is $\top$.

Definition 2 (Forgetting). *$Fg(p)$ is the event model $((\mathsf{S}, \mathsf{R}, \mathsf{pre}, \mathsf{post}), \mathsf{S}')$ where $\mathsf{S} = \{0, 1\}$, $\mathsf{R} = \mathsf{S} \times \mathsf{S}$, $\mathsf{pre}(0) = \top$ and $\mathsf{pre}(1) = \top$, $\mathsf{post}(0)(p) = \bot$ and $\mathsf{post}(1)(p) = \top$ (and $\mathsf{post}(i)(q) = q$ for all $q \neq p$, $i = 0, 1$), and $\mathsf{S}' = \mathsf{S}$.*

Definition 3 (Semantics). *Assume an epistemic model $M = (S, R, V)$.*

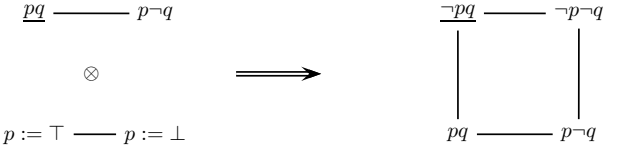
$$
\begin{array}{lll}
M, s \models p & \textit{iff} & s \in V(p) \\
M, s \models \neg\varphi & \textit{iff} & M, s \not\models \varphi \\
M, s \models \varphi \wedge \psi & \textit{iff} & M, s \models \varphi \textit{ and } M, s \models \psi \\
M, s \models K\varphi & \textit{iff} & \textit{for all } t \in S : (s, t) \in R \textit{ implies } M, t \models \varphi \\
M, s \models [Fg(p)]\varphi & \textit{iff} & M \otimes Fg(p), (s, 0) \models \varphi \textit{ and } M \otimes Fg(p), (s, 1) \models \varphi
\end{array}
$$

where $M \otimes Fg(p) = (S', R', V')$ such that $S' = S \times \{0, 1\}$, $((s, i), (t, j)) \in R'$ iff $(s, t) \in R$ and $i, j \in \{0, 1\}$, $V'(p) = \{(s, 1) \mid s \in S\}$ and $V'(q) = V(q) \times S$ for $q \neq p$. The set of validities is called FG.

In fact, $M \otimes Fg(p)$ is the restricted modal product of M and event model $Fg(p)$ according to [19,17], which in this case amounts to taking two copies of the model M, making p true everywhere in the first, making p false everywhere in the second, and making corresponding states indistinguishable for the agent.

Example. We visualize $S5$ models by linking states that are indistinguishable for an agent. Reflexivity and transitivity are assumed. In these visualizations we abuse the language by writing valuations instead of states and postconditions instead of events, and we write $\top$ for an event with empty postcondition. The actual state is underlined.

Suppose the agent knows p but does not know q (and where in fact q is true), and where the agent forgets that p. The execution of event model $Fg(p)$ (and where in fact p becomes false) is pictured as follows. In the resulting Kripke model, the agent no longer knows p, and remains uncertain about q.



Deterministic forgetting. The pointed (deterministic) versions of the forgetting event can be defined as notational abbreviations of the not-pointed primitives $[Fg(p), 1]\varphi \equiv_{\mathsf{def}} [Fg(p)](p \to \varphi)$ and $[Fg(p), 0]\varphi \equiv_{\mathsf{def}} [Fg(p)](\neg p \to \varphi)$. From this follow the validities $\langle Fg(p), 0\rangle\varphi \leftrightarrow \langle Fg(p)\rangle(\neg p \wedge \varphi)$ and $\langle Fg(p), 1\rangle\varphi \leftrightarrow \langle Fg(p)\rangle(p \wedge \varphi)$, and also the axiom for non-determinism

$$[Fg(p)]\varphi \leftrightarrow [Fg(p), 0]\varphi \wedge [Fg(p), 1]\varphi.$$

Axiomatization. To obtain a complete axiomatization **FG** for the logic FG we apply the reduction axioms for event models, as specified in [19] and [17]. The case $[Fg(p)]p$ for the forgotten atom expresses that you cannot guarantee that p is true after forgetting it *by way of varying its value*; see Section 4 for a modelling where actual facts do not change value after forgetting. In the case for negation, note that $[Fg(p)]\neg\varphi$ is *not* equivalent to $\neg[Fg(p)]\varphi$, and note the correspondence with deterministic forgetting by abbreviation. The epistemic operator commutes with the forgetting operator. Thus the consequences of forgetting are known before it takes place ('no miracles'). We emphasize that the negated epistemic operator (for 'possible that') does *not* commute with forgetting ($[Fg(p)]\neg K\varphi$ is *not* equivalent to $\neg K[Fg(p)]\varphi$); therefore, K cannot be eliminated. Further details are omitted.[2] It follows that the axiomatization **FG** is sound and complete.

Definition 4 (Axiomatization FG). *Only axioms involving Fg are shown.*

$$\begin{array}{lcll}
[Fg(p)]p & \leftrightarrow & \bot & \\
[Fg(p)]q & \leftrightarrow & q & \text{for } q \neq p \\
[Fg(p)]\neg\varphi & \leftrightarrow & \neg[Fg(p)](\neg p \to \varphi) \wedge \neg[Fg(p)](p \to \varphi) & \\
[Fg(p)](\varphi \wedge \psi) & \leftrightarrow & [Fg(p)]\varphi \wedge [Fg(p)]\psi & \\
[Fg(p)]K\varphi & \leftrightarrow & K[Fg(p)]\varphi &
\end{array}$$

Theorem 1. *Axiomatization* **FG** *is sound and complete.*

Proof. The axiomatization resulted from applying the reduction axioms in [19,17]. This kills two birds (soundness and completeness) in one throw. We show the basic case for the forgotten atom (the relevant axiom is $[\mathsf{M}, \mathsf{s}]p \leftrightarrow (\mathsf{pre}(\mathsf{s}) \to \mathsf{post}(\mathsf{s})(p))$ and non-determinism).

[2] The axiomatization **FG** can be made into a *reduction* system by having pointed event models as primitives instead of abbreviations. We then employ the reduction axiom for non-determinism (above) and $[Fg(p), 0]\neg\psi \leftrightarrow (\neg p \to [Fg(p), 0]\psi)$, etc.

$[Fg(p)]p$
iff
$[Fg(p), 0]p \wedge [Fg(p), 1]p$
iff
$(\mathsf{pre}(0) \to \mathsf{post}(0)(p)) \wedge (\mathsf{pre}(1) \to \mathsf{post}(1)(p))$
iff
$(\top \to \bot) \wedge (\top \to \top)$
iff
$\bot$

4 Results, and Other Forgetting Operators

Using this simple logic we can now harvest an interesting number of theoretical results. A number of different perspectives on forgetting propositional variables (such as release, elimination, bisimulation quantification, symmetric contraction, and value swapping or switching) all amount to the same: although resulting in different structures, these cannot be distinguished from each other in the language, i.e., they all represent the same set of believed formulas. An important Theorem 2 states that becoming unaware (the original [1]-sense of forgetting as data abstraction) is the same as becoming ignorant. We also show that our results generalize to more agents or variables. (As long as R is serial, so that $K\bot \leftrightarrow \bot$ and $K\top \leftrightarrow \top$,) Ignorance is indeed obtained (and, trivially also awareness of it—$[Fg(p)]K(\neg Kp \wedge \neg K\neg p)$):

Proposition 1. *$[Fg(p)](\neg Kp \wedge \neg K\neg p)$ is valid.*

Forgetting without changing the real world. An unfortunate side effect of our modelling of forgetting is that the actual value of p gets lost in the process of forgetting, such as in the example on page 22. This is undesirable if we *only* want to model that the agents forget the value of p but that otherwise nothing changes: in particular, the actual value of p should not change. We can overcome that deficiency in the event model for *epistemic forgetting*.

Definition 5 (Epistemic forgetting). *Epistemic forgetting is the pointed event model $(\mathsf{Fg}(p), n)$ where $\mathsf{Fg}(p)$ is like $Fg(p)$ except that there is one more event n in the model, indistinguishable from the other two, with empty postcondition (and with precondition $\top$).*

The point n represents the event that 'nothing happens'. As it is the point of the event model, it ensures that the actual value of p does not change. We can visualize this event as

$$p := \top \text{ ——— } \underline{\top} \text{ ——— } p := \bot$$

Definition 6 (Axioms for epistemic forgetting). *The axioms for $(\mathsf{Fg}(p), n)$ are as for $Fg(p)$ except that*

$$\begin{array}{lcl} [\mathsf{Fg}(p), n]p & \leftrightarrow & p \\ [\mathsf{Fg}(p), n]\neg\varphi & \leftrightarrow & \neg[\mathsf{Fg}(p), 0]\varphi \wedge \neg[\mathsf{Fg}(p), 1]\varphi \wedge \neg[\mathsf{Fg}(p), n]\varphi \end{array}$$

Dynamic modal operators for pointed events $(\mathsf{Fg}(p), 0)$ and $(\mathsf{Fg}(p), 1)$ are again introduced in the language by abbreviation, now from $[\mathsf{Fg}(p), n]$, somewhat different from before. We have the results that (proof omitted)

Proposition 2 (Preservation of factual information).
Schema $\psi \to [\mathsf{Fg}(p), n]\psi$ *is valid for boolean* ψ.

Proposition 3 (Epistemic propositions are preserved).
Schema $[\mathsf{Fg}(p), n]K\psi \leftrightarrow [Fg(p)]K\psi$ *is valid (for all* ψ *in the language).*

In other words, from the perspective of the agent, the different modellings of forgetting are indistinguishable. The different occurrences of ψ in Proposition 3 are actually in different languages (both may contain forgetting operators!), a trivial translation can make this more precise.

That makes the simpler modelling $Fg(p)$ preferable over the slightly more complex $(\mathsf{Fg}(p), n)$. We are vague in Proposition 3 about 'the language', as the language with $[\mathsf{Fg}(p), n]\varphi$ as inductive construct is different from the language with $[Fg(p)]\varphi$ as inductive construct. More strictly the result is that $[\mathsf{Fg}(p), n]K\psi \leftrightarrow [Fg(p)]K\mathsf{trs}(\psi)$ is valid, subject to the translation with inductive clause $\mathsf{trs}([\mathsf{Fg}(p), n]\varphi) = [Fg(p)]\mathsf{trs}(\varphi)$.

Swapping values. Yet another way to model forgetting is by making every state in the model indistinguishable from one wherein the value of p has been swapped / switched: if true, it became false, and if false it became true.

Definition 7 (Forgetting by swapping values). *Forgetting by swapping is the pointed event model that is like* $Fg(p)$ *except that in one event, the actual event, nothing happens, whereas in the other event the assignment* $p := \neg p$ *is executed.*

$$p := \neg p \quad \underline{\top}$$

Again we can adjust the axiomatization, we obtain the results that actual facts do not change value, and that propositions under the scope of the epistemic operator are preserved.

Scrambling the valuation of the forgotten atom. Instead of making p randomly (but indistinguishably!) true or false in every state of the Kripke model, the more proper way of 'releasing the value of p' in a modal logical context is to make p randomly true in a subset of the domain of the model. One can then make all those results indistinguishable from one another for the agent. Unlike the former, where two copies of the model M suffice, we now need $2^{|M|}$ copies.

Consider again the structure pq——$p\neg q$ encoding that the agent knows p but is ignorant about q. In proper Lin and Reiter [1] fashion, the models agreeing with pq——$p\neg q$ on anything except maybe p are the following four:

$$pq\text{——}p\neg q \quad pq\text{——}\neg p\neg q \quad \neg pq\text{——}p\neg q \quad \neg pq\text{——}\neg p\neg q$$

These four still have ignorance about q in common, but only two of them satisfy ignorance about p. We have encoded *unawareness* of p, but not *ignorance* about

p. If we make corresponding points in all these models indistinguishable for the agent, it again follows that the agent is ignorant about p, and the result is bisimilar to that achieved by $Fg(p)$. (Bisimilarity is a notion of structural similarity that guarantees logical equivalence, i.e., of sets of beliefs, see [20].) Doing so, we can after all reclaim *unawareness* of p, using the more abstract perspective of *bisimulation quantification*: apart from the four above, pq——$p\neg q$ is also similar except for the value of p to other structures, e.g. to pq——$p\neg q$——$\neg pq$ (three indistinguishable states), which satisfies that $K(p \vee q)$ is true. Because if we abstract from the value of p, q——$\neg q$ is bisimilar to q——$\neg q$——q. This prepares the ground for the next paragraph.

Bisimulation quantification. Becoming unaware of an atom p can be modelled as universal bisimulation quantification over p [21,22,23] namely as

$$[Fg^{\forall}(p)]\varphi \equiv_{\mathsf{def}} \forall p\varphi$$

where $M, s \models \forall p\varphi$ iff for all (M', s') s.t. $(M', s') \leftrightarrows_{P-p} (M, s) : (M', s') \models \varphi$.

The notation $(M', s') \leftrightarrows_{P-p} (M, s)$ means that epistemic state (M', s') is bisimilar to epistemic state (M, s) with respect to the set of all atoms *except* p.[3] In other words the valuation of p may vary 'at random'. This includes the model constructed by $Fg(p)$ (and that by $(\mathsf{Fg}(p), n)$) from a given M so that

$$M \leftrightarrows_{P-p} M \otimes Fg(p)$$

which immediately delivers that:

Theorem 2. *If $\psi \in \mathcal{L}(P - p)$ then $\psi \rightarrow [Fg(p)]\psi$ is valid.*

This fixes *progression* in the AI sense (which formulas initially believed that do not involve p are still believed in the new state where p is forgotten), and therefore creates a strong link with [2].

Of course we do not have that $[Fg^{\forall}(p)](\neg Kp \wedge \neg K\neg p)$ is valid. To adjust this 'becoming unaware of p' towards 'becoming ignorant of p', we have to let all $P - p$-bisimilar states be indistinguishable by the agent.[4] For this alternative 'becoming ignorant by bisimulation quantification' operation $\mathsf{Fg}^{\forall}(p)$ we again have the desired $[\mathsf{Fg}^{\forall}(p)](\neg Kp \wedge \neg K\neg p)$ and we then also have that $[\mathsf{Fg}^{\forall}(p)]K\varphi \leftrightarrow [Fg(p)]K\varphi$. The much simpler $Fg(p)$ is preferable for computational reasons.

Multiple variables. The forgetting of multiple variables can be modelled by a simple adjustment. For n propositional variables, we get an event model

[3] Applied to forgetting, this is the original proposal in [4]. Then, to achieve ignorance as in [3] they constrain this set of models to those satisfying $\neg Kp \wedge \neg K\neg p$. That is different from what we do.

[4] Given (M, s), let $\mathfrak{M} = \{(M\ , s\ ) \mid (M, s) \leftrightarrows_{P\ \ p} (M\ , s\ )\}$. Given $\mathfrak{R} : (M, s) \leftrightarrows_{P\ \ p} (M\ , s\ )$ and $\mathfrak{R}\ : (M, s) \leftrightarrows_{P\ \ p} (M\ \ , s\ )$, add pairs $(s\ , s\ )$ to the relation R on $\mathfrak{M}$ whenever there is a $s \in S$ such that $(s, s\ ) \in \mathfrak{R}$ and $(s, s\ ) \in \mathfrak{R}$. Then $\mathfrak{M} \models \neg Kp \wedge \neg K\neg p$.

$Fg(p_1, \ldots, p_n)$ with a domain consisting of 2^n events, one for each combination of assignments of different variables to true and false. All prior results still follow (including bisimulation quantification for n variables).

Combining learning and forgetting. One might wish to combine forgetting with other dynamic operations such as learning (by public announcements). We simply add an inductive construct $[\varphi]\psi$ to the language, which stands for 'after announcement of φ, ψ holds' (see [24]). The resulting logic is again equally expressive as epistemic logic: just add the rewrite rules involving announcements.

Multiple agents. Given a parameter set of agents A, we adjust the language to $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_a\varphi \mid [Fg_B(p)]\varphi$, where $a \in A$ and $B \subseteq A$, and we adjust the accessibility relation $R \subseteq (S \times S)$ to an accessibility function $R : A \to \mathcal{P}(S \times S)$—accessibility relation R_a (for $R(a)$) interprets operator K_a (knowledge of agent a). (Further details omitted.) The case $Fg_A(p)$ where $B = A$ models forgetting as group ignorance, and the case $Fg_a(p)$ where $B = \{a\}$ models the forgetting of an individual a in the presence of A (see page 19). Both are most succinctly modelled by a version of 'swapping values' forgetting (Definition 7) namely as event models visualized as

$$p := \neg p \xrightarrow{\quad A \quad} \top \qquad\qquad p := \neg p \xrightarrow{\quad a \quad} \top$$

The visualization on the right means that all agents *except* a can distinguish between the two alternatives: access for a is the universal relation on the domain, and access for all agents in $A - a$ is the identity. Again, all former results generalize, both versions are axiomatizable very similarly to the previous, etc. The more obvious multi-agent version of $Fg(p)$ (with assignments to true and to false only) does *not* model individual forgetting in a group: this would express that the other agents learn that p is true or learn that p is false, clearly undesirable.

5 Further Research

Remembering prior knowledge. For the agent to recall prior knowledge we have to be able to refer to past events. Let $Fg(p)^-$ be the converse of $Fg(p)$ (e.g. in the sense of [13,14,15]). Awareness of present ignorance and prior knowledge about p can now be formalized as

$$K(\neg Kp \wedge \neg K\neg p \wedge \langle Fg(p)^-\rangle(Kp \vee K\neg p))$$

We now need a structure allowing us to interpret such converse events. This is not possible in pointed Kripke models, but it can be elegantly done employing what is known as the 'forest' produced by the initial Kripke model and all possible sequences of all $Fg(p)$ events (for all atoms), see [25,26,27,28,14,15]. We now add assignments to the language, as in the underlying proposal, and additionally add theories for event models using converse actions [13,26]. Thus we get a complete axiomatization, though not a reduction result to epistemic formulas (converse events cannot be eliminated from the logical language by equivalences).

Regression and progression. By applying equivalences we can reduce a formula of the form $[Fg(p)]\psi$ to an equivalent expression χ without dynamic operators—ψ is a final condition and χ is an initial condition that is derived from it by way of these equivalences. This process is known as *regression*. Dynamic epistemic logic is very suitable for this kind of regression, and there are efficient model checkers for epistemic formulas. Progression is harder in this setting. (See page 20.)

Forgetting modal formulas. How to model the forgetting modal formulas is a different piece of cake altogether; in this case we have made no progress yet.

Forgetting of events. This amounts to introducing *temporal uncertainty* in the model, apart from epistemic uncertainty. This can be done by introducing histories of events to structures, or moving to a temporal epistemic perspective using 'forests', as above, see [26]. It is clear how this has to be done, and the results should prove interesting.

Acknowledgements. The authors would like to thank the four anonymous AI08 referees, and also the participants of the ESSLLI 2008 Hamburg workshop Logic and Intelligent Interaction. Hans van Ditmarsch acknowledges support of the Netherlands Institute of Advanced Study where he was Lorentz Fellow in 2008.

References

1. Lin, F., Reiter, R.: Forget it! In: AAAI Fall Symposium on Relevance, New Orleans (1994)
2. Lang, J., Liberatore, P., Marquis, P.: Propositional independence: Formula-variable independence and forgetting. J. Artif. Intell. Res. (JAIR) 18, 391–443 (2003)
3. Baral, C., Zhang, Y.: Knowledge updates: semantics and complexity issues. Artificial Intelligence 164(1-2), 209–243 (2005)
4. Zhang, Y., Zhou, Y.: Knowledge forgetting: Properties and applications. Work in progress under submission (2008)
5. Wang, K., Sattar, A., Su, K.: A theory of forgetting in logic programming. In: AAAI, pp. 682–688 (2005)
6. Zhang, Y., Foo, N., Wang, K.: Solving logic program conflict through strong and weak forgettings. In: Proceedings of IJCAI, pp. 627–634 (2005)
7. Eiter, T., Wang, K.: Forgetting and conflict resolving in disjunctive logic programming. In: Proceedings of AAAI (2006)
8. Zhao, Y., Wang, K., Topor, R., Pan, J., Giunchiglia, F.: Semantic cooperation and knowledge reuse by using autonomous ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 666–679. Springer, Heidelberg (2007)
9. Erdem, E., Ferraris, P.: Forgetting actions in domain descriptions. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, pp. 409–414. AAAI Press, Menlo Park (2007)
10. Herzig, A., Lang, J., Marquis, P.: Action representation and partially observable planning using epistemic logic. In: Proceedings of IJCAI (2003)

11. van Ditmarsch, H.: Prolegomena to dynamic logic for belief revision. Synthese (Knowledge, Rationality & Action) 147, 229–275 (2005)
12. Baltag, A., Smets, S.: Dynamic belief revision over multi-agent plausibility models. In: Proceedings of LOFT 2006 (7th Conference on Logic and the Foundations of Game and Decision Theory) (2006)
13. Aucher, G.: Perspectives on belief and change. PhD thesis, University of Otago & Institut de Recherche en Informatique de Toulouse, New Zealand & France (2008)
14. Yap, A.: Product update and looking backward. Technical report, University of Amsterdam (2006); ILLC Research Report PP-2006-39
15. Sack, Y.: Adding Temporal Logic to Dynamic Epistemic Logic. PhD thesis, Indiana University, Bloomington, USA (2007)
16. van Benthem, J., van Eijck, J., Kooi, B.: Logics of communication and change. Information and Computation 204(11), 1620–1662 (2006)
17. van Ditmarsch, H., Kooi, B.: Semantic results for ontic and epistemic change. In: Bonanno, G., van der Hoek, W., Wooldridge, M. (eds.) Post-proceedings of LOFT 2006, Amsterdam University Press (2008) (to appear in the series Texts in Logic and Games)
18. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. Foundations of Computing Series. MIT Press, Cambridge (2000)
19. Baltag, A., Moss, L., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. In: Gilboa, I. (ed.) Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 1998), pp. 43–56 (1998)
20. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press, Cambridge (2001)
21. Visser, A.: Bisimulations, model descriptions and propositional quantifiers, Logic Group Preprint Series 161, Department of Philosophy, Utrecht University (1996)
22. French, T.: Bisimulation quantifiers for modal logic. PhD thesis, University of Western Australia (2006)
23. van Ditmarsch, H., French, T.: Simulation and information. In: (Electronic) Proceedings of LOFT 2008, Amsterdam (2008)
24. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Synthese Library, vol. 337. Springer, Heidelberg (2007)
25. Ramanujam, R., Lodaya, K.: Proving Fairness of Schedulers. In: Parikh, R. (ed.) Logic of Programs 1985. LNCS, vol. 193, pp. 284–301. Springer, Heidelberg (1985)
26. van Benthem, J., Gerbrandy, J., Pacuit, E.: Merging frameworks for interaction: DEL and ETL. In: Samet, D. (ed.) Proceedings of TARK 2007, pp. 72–81 (2007)
27. Pacuit, E.: Some comments on history-based structures. Journal of Applied Logic (2007)
28. van Ditmarsch, H., Ruan, J., van der Hoek, W.: Model checking dynamic epistemics in branching time. In: (Informal) Proceedings of FAMAS 2007, Durham, UK (2007)

A Fixed-Point Property of Logic-Based Bargaining Solution

Dongmo Zhang

Intelligent Systems Laboratory
University of Western Sydney, Australia
dongmo@scm.uws.edu.au

Abstract. This paper presents a logic-based bargaining solution based on Zhang and Zhang's framework. It is shown that if the demand sets of players are logically closed, the solution satisfies a fixed-point property, which says that the outcome of bargaining is the result of mutual belief revision. The result is interesting not only because it presents a desirable logical property of bargaining solution but also establishes a link between bargaining theory and multi-agent belief revision.

1 Introduction

Negotiation or bargaining is a process of dispute resolution to reach mutually beneficial agreements. The studies of negotiation in game theory, known as *bargaining theory*, initiated by John Nash's path-breaking work [1], has reached a high sophistication with a variety of models and solutions and has been extensively applied to economics, sociology, management science, and politics [2,3,4].

The game-theoretical model of bargaining is purely numerical. Although the numerical theory of bargaining provides "a 'clear-cut' numerical predication for a wide range of bargaining problems", it does not help us to understand how disputes are resolved through a bargaining process ([5] p.81-88).

In recent years, the AI researchers try to rebuild the theory of bargaining and negotiation in order to model logical reasoning behind a bargaining process. Kraus *et al.* introduced a logical model of negotiation based on argumentation theory [6,7]. Unlike game theory, the model allows explicit representation of negotiation items, promises, threats and arguments. More importantly, bargaining process can be embedded into logic-based multi-agent systems so that negotiation becomes a component of agent planning. Similar to Rubinstein's strategic model of bargaining, the argumentation-based approach views bargaining as a non-cooperative game. Zhang *et al.* introduced a logical model of negotiation based on belief revision theory [8,9,10]. Different from the argumentation-based framework, the belief-revision-based approach takes a cooperative view. In order to reach an agreement, each player tries to persuade the other player to accept her demands or beliefs. Anyone who is convinced to accept the other player's demands will need to conduct a course of belief revision. It was assumed that

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 30–41, 2008.

any possible outcome of negotiation, (Ψ_1, Ψ_2), should satisfies the following fixed-point condition [11], which says that the outcome of negotiation is the common demands or beliefs after mutual belief revision:

$$Cn(\Psi_1 \cup \Psi_2) = (Cn(X_1) \otimes_1 \Psi_2) \cap (Cn(X_2) \otimes_2 \Psi_1)$$

where X_i contains the demands of agent i and $\otimes_i$ is the belief revision operator of agent i. However, there is no justification for the assumption. This paper aims to build a concrete bargaining solution to satisfy the fixed-point condition. The construction of the solution is based on the bargaining model proposed by Zhang and Zhang in [12,13]. The result of the paper not only shows the logical property of bargaining but also establishes the link between bargaining and belief revision, which may be helpful for the investigation of multi-agent belief revision.

2 Logical Model of Bargaining

Within this paper, we consider the bargaining situations with two players. We assume that each party has a set of negotiation items, referred to as *demand set*, described by a finite propositional language $\mathcal{L}$. The language is that of classical propositional logic with an associated consequence operation Cn in the sense that $Cn(X) = \{\varphi : X \vdash \varphi\}$, where X is a set of sentences. A set X of sentences is *logically closed* or called a *belief set* when $X = Cn(X)$. If X and Y are two sets of sentences, $X + Y$ denotes $Cn(X \cup Y)$.

Suppose that X_1 and X_2 are the demand sets from two bargaining parties respectively. To simplify exploration, we use X_{-i} to represent the other set among X_1 and X_2 if X_i is one of them. If D is a vector of two components, D_1 and D_2 will represent each of the components of D.

2.1 Bargaining Games

We will use the bargaining model introduced by Zhang and Zhang in [12] to represent a bargaining situation.

Definition 1. [12] *A bargaining game is a pair* $((X_1, \succeq_1), (X_2, \succeq_2))$*, where* X_i $(i = 1, 2)$ *is a logically consistent set of sentences in* $\mathcal{L}$ *and* $\succeq_i$ *is a complete transitive reflexive order (total preorder or weak order) over* X_i *which satisfies the following logical constraints*[1]*:*

(LC) *If* $\varphi_1, \cdots, \varphi_n \vdash \psi$*, then there is* k $(1 \leq k \leq n)$ *such that* $\psi \succeq_i \varphi_k$*.*

[1] A complete transitive reflexive order, i.e., total preorder or weak order, satisfies the following properties:

- Completeness or totality: $\varphi \preceq \psi$ or $\psi \preceq \varphi$.
- Reflexivity: $\varphi \preceq \varphi$.
- Transitivity: if $\varphi \preceq \psi$ and $\psi \preceq \chi$ then $\varphi \preceq \chi$.

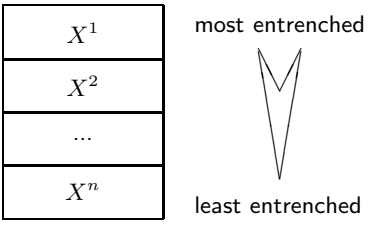


Fig. 1. The hierarchy of a demand set

We call the pair $(X_i, \succeq_i)$ *the prioritized demand set of player* i*. For any* $\varphi, \psi \in X_i$*,* $\psi \succ \varphi$ *denotes that* $\psi \succeq_i \varphi$ *and* $\varphi \not\succeq_i \psi$*.* $\psi \approx_i \varphi$ *denotes that* $\psi \succeq_i \varphi$ *and* $\varphi \succeq_i \psi$*.*

Intuitively, a bargaining game is a formal representation of a bargaining situation whereby each player describes his demands in logical formulae and expresses his preferences over his demands in total preorder. We assume that each player has consistent demands. The preference ordering of each player reflects the *degree of entrenchment* in which the player defends his demands. The logical constraint (LC) says that if $\varphi_1, \cdots, \varphi_n$ and ψ are all your demands and $\varphi_1, \cdots, \varphi_n \vdash \psi$, then ψ should not be less entrenched than all the φ_i because if you fail to defend ψ, at least one of the φ_i has to be dropped (otherwise you would not have lost ψ). This indicates that the preference orderings are different from players' payoff or utility. For instance, suppose that p_1 represents the demand of a seller "*the price of the good is no less than \$10*" and p_2 denotes "*the price of the good is no less than \$8*". Obviously the seller could get higher payoff from p_1 than p_2. However, since p_1 implies p_2, she will entrench p_2 no less than p_1, *i.e.*, $p_2 \succeq p_1$, because, if she fails to keep p_1, she can still bargain for p_2 but the loss of p_2 means the loss of both.

Given a prioritized demand set $(X, \succeq)$, we define recursively a hierarchy, $\{X^j\}_{j=1}^{+\infty}$, of X with respect to the ordering $\succeq$ as follows:

1. $X^1 = \{\varphi \in X : \neg\exists\psi \in X(\psi \succ \varphi)\}$; $T^1 = X \backslash X^1$.
2. $X^{j+1} = \{\varphi \in T^j : \neg\exists\psi \in T^j(\psi \succ \varphi)\}$; $T^{j+1} = T^j \backslash X^{j+1}$.

where $\psi \succ \varphi$ denotes $\psi \succeq \varphi$ and $\varphi \not\succeq \psi$. The intuition behind the construction is that, at each stage of the construction, we collects all maximal elements from the current demand set and remove them from the set for the next stage of the construction. It is easy to see that there exists a number n such that $X = \bigcup_{j=1}^{n} X^j$ due to the logical constraint LC[2].

It is easy to see that for any $\varphi \in X^j$ and $\psi \in X^k$, $\varphi \succ \psi$ if and only if $j < k$. In the sequel, we write $X^{\leq k}$ to denote $\bigcup_{j=1}^{k} X^j$.

[2] Note that X can be an infinite set even though the language is finite.

Based on the hierarchy of each demand, we can define a belief revision function for each agent by following Nebel's idea of *prioritized base revision* [14]:

For any demand set $(X, \preceq)$ and a set, F, of sentences,

$$X \otimes F \stackrel{def}{=} \bigcap_{H \in X \Downarrow F} (H + F),$$

where $X \Downarrow F$ is defined as: $H \in X \Downarrow F$ if and only if

1. $H \subseteq X$,
2. for all k $(k = 1, 2, \cdots)$, $H \cap X^k$ is a maximal subset of X^k such that $\bigcup_{j=1}^{k} (H \cap X^j) \cup F$ is consistent.

In other words, H is a maximal subset of X that is consistent with F and gives priority to the higher ranked items. The following result will be used in Section 3.

Lemma 1. [14] *If X is logically closed, then $\otimes$ satisfies all AGM postulates.*

2.2 Possible Agreements

Similar to [12], we define a possible outcome of negotiation as a concession made by two players.

Definition 2. *Let $G = ((X_1, \succeq_1), (X_2, \succeq_2))$ be a bargaining game. A deal of G is a pair (D_1, D_2) satisfying the following conditions: for each $i = 1, 2$,*

1. *$D_i \subseteq X_i$;*
2. *$X_1 \cap X_2 \subseteq D_i$;*
3. *for each k $(k = 1, 2, \cdots)$, $D_i \cap X_i^k$ is a maximal subset of X_i^k such that $\bigcup_{j=1}^{k} (D_i \cap X_i^j) \cup D_{-i}$ is consistent.*

where $\{X_i^j\}_{k=1}^{+\infty}$ is the hierarchy of X_i. The set of all deals of G is denoted by $\Omega(G)$, called the feasible set of the game.

Intuitively, a possible agreement is a pair of subsets of two players' original demand sets such that the collection of remaining demands is consistent. Obviously each player would like to keep as many original demands as possible. Therefore, if a player has to give up a demand, the player typically gives up the ones with the lowest priority. Note that we require that no player gives up common demands, which is crucial to the fixed-point property. This is different from Zhang and Zhang's definition in [12].

2.3 Bargaining Solution

We have shown how to generates all possible deals from a bargaining game. However, a game might have multiple deals. Different deals would be in favor of

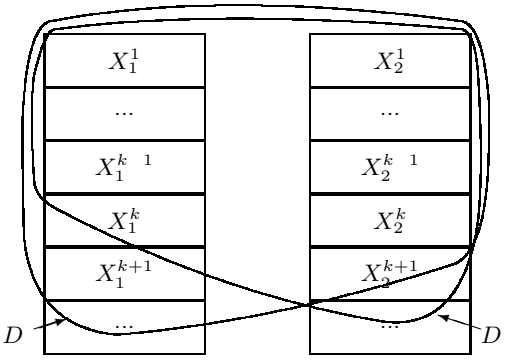


Fig. 2. Different deals are in favour of different parties

different parties. The major concern of a bargaining theory is how to measure and balance the gain of each negotiating party.

Instead of counting the number of demands a deal contains for each party, we consider the top block demands a player keeps in the deal (the top levels of demands in each player's demand hierarchy) and ignore all demands that are not included in the top blocks except the common demands[3].

Given a deal D, we shall use the maximal top levels of each player's demands the deal contains as the indicator of the player's gain from the deal, i.e., $\max\{k : X_i^{\leq k} \subseteq D_i\}$. For instance, in Figure 2, player 1 can successfully remain maximally top $k+1$ levels of his demands from deal D' while player 2 gains maximally top k levels of his demands from the deal.

To compare players' gains from different deals, we use the gain of the player with smaller gain from a deal as the index of the deal, i.e., $\min\{\max\{k : X_1^{\leq k} \subseteq D_1\}, \max\{k : X_2^{\leq k} \subseteq D_2\}\}$, or equivalently, $\max\{k : X_1^{\leq k} \subseteq D_1 \text{ and } X_2^{\leq k} \subseteq D_2\}$. For instance, in Figure 2, the gain index of D' is k while the gain index of D'' is $k-1$. By using this index, we can collect all the best deals of a game:

$$\gamma(G) = \arg \max_{(D_1,D_2)\in\Omega(G)} \{k : X_1^{\leq k} \subseteq D_1 \text{ and } X_2^{\leq k} \subseteq D_2\}$$

Based on the intuitive description, we are now ready to construct our bargaining solution.

Definition 3. *A bargaining solution is a function F which maps a bargaining game $G = ((X_1, \succeq_1), (X_2, \succeq_2))$ to a pair of sets of sentences defined as follows:*

$$F(G) \stackrel{def}{=} \Big(\bigcap_{(D_1,D_2)\in\gamma(G)} D_1, \bigcap_{(D_1,D_2)\in\gamma(G)} D_2\Big) \tag{1}$$

where $\gamma(G) = \arg \max_{(D_1,D_2)\in\Omega(G)} \{k : X_1^{\leq k} \subseteq D_1 \text{ and } X_2^{\leq k} \subseteq D_2\}$.

[3] Note that common demands of two parties are always included in a deal no matter how much priorities they have.

Let

$$\pi^G_{max} = \max_{(D_1,D_2)\in\Omega(G)} \{k : X_1^{\leq k} \subseteq D_1 \text{ and } X_2^{\leq k} \subseteq D_2\} \tag{2}$$

and

$$(\Phi_1, \Phi_2) = (X_1^{\leq \pi^G_{max}}, X_2^{\leq \pi^G_{max}}) \tag{3}$$

We call $\Phi = (\Phi_1, \Phi_2)$ the *core of the game*. Intuitively, the core of the game is the pair of maximal top block demands that are contained in all the best deals.

To help the reader to understand our solution, let us consider the following example.

Example 1. *A couple are making their family budget for the next year. The husband wants to change his car to a new fancy model and have a domestic holiday. The wife is going to implement her dream of a romantic trip to Europe and suggests to redecorate the kitchen. Both of them know that they can't have two holidays in one year. They also realize that they cannot afford a new car and an overseas holiday in the same year without getting a loan from the bank. However, the wife does not like the idea of borrowing money.*

In order to represent the situation in logic, let c denote "buy a new car", d stand for "domestic holiday", o for "overseas holiday", k for "kitchen redecoration" and l for "loan". Then $\neg(d \wedge o)$ means that it is impossible to have both domestic holiday and overseas holiday. The statement $(c \wedge o) \rightarrow l$ says that if they want to buy a new car and also have an overseas holiday, they have to get a loan from the bank.

With the above symbolization, we can express the husband's demands in the following set:

$$X_1 = \{c, d, \neg(d \wedge o), (c \wedge o) \rightarrow l\}$$

Similarly, the wife's demands can be represented by:

$$X_2 = \{o, k, \neg(d \wedge o), (c \wedge o) \rightarrow l, \neg l\}$$

Assume that the husband's preferences over his demands are:

$$\neg(d \wedge o) \approx_1 (c \wedge o) \rightarrow l \succ_1 c \succ_1 d$$

and the wife's preferences are:

$$\neg(d \wedge o) \approx_2 (c \wedge o) \rightarrow l \succ_2 o \succ_2 k \succ_2 \neg l$$

Let G represent the bargaining game. It is easy to calculate that the game has the following three possible deals:

$D^1 = (\{\neg(d \wedge o), (c \wedge o) \rightarrow l, c, d\}, \{\neg(d \wedge o), (c \wedge o) \rightarrow l, k, \neg l\})$.
$D^2 = (\{\neg(d \wedge o), (c \wedge o) \rightarrow l, c\}, \{\neg(d \wedge o), (c \wedge o) \rightarrow l, o, k\})$.
$D^3 = (\{\neg(d \wedge o), (c \wedge o) \rightarrow l\}, \{\neg(d \wedge o), (c \wedge o) \rightarrow l, o, k, \neg l\})$.

The core of the game is then:

$$(\{\neg(d \wedge o), (c \wedge o) \rightarrow l, c\}, \{\neg(d \wedge o), (c \wedge o) \rightarrow l, o\})$$

$\gamma(G)$ contains only a single deal, which is D^2. The solution is then

$$F(G) = D^2 = (\{\neg(d \wedge o), (c \wedge o) \rightarrow l, c\}, \{\neg(d \wedge o), (c \wedge o) \rightarrow l, o, k\})$$

In words, the couple agree upon the commonsense that they can only have one holiday and they have to get a loan if they want to buy a new car and to go overseas for holiday. The husband accepts his wife's suggestion to have holiday in Europe and the wife agrees on buying a new car. As a consequence of the agreement, they agree on getting a loan to buy the car.

3 Fixed Point Property

In [11], it was argued that a procedure of negotiation can be viewed as a course of mutual belief revision when players' belief states with respect to the negotiation are specified by the demand sets of the bargaining game.

Before we show the fixed-point property of the solution we construct, let us consider two facts on the solution:

Lemma 2. $\pi^G_{max} = \max\{k : X_1^{\leq k} \cup X_2^{\leq k} \cup (X_1 \cap X_2)$ *is consistent*$\}$.

Proof. Let $\pi = \max\{k : X_1^{\leq k} \cup X_2^{\leq k} \cup (X_1 \cap X_2)$ is consistent$\}$. It is easy to show that $X_1^{\leq \pi^G_{max}} \cup X_2^{\leq \pi^G_{max}} \cup (X_1 \cap X_2)$ is consistent because $\gamma(G)$ is non-empty. Therefore $\pi^G_{max} \leq \pi$. On the other hand, since $X_1^{\leq \pi} \cup X_2^{\leq \pi} \cup (X_1 \cap X_2)$ is consistent, there exists a deal $(D_1, D_2) \in \Omega(G)$ such that $X_i^{\leq \pi} \subseteq D_i$ and $X_1 \cap X_2 \subseteq D_i$ for each $i = 1, 2$. Thus $\pi \leq \pi^G_{max}$. We conclude that $\pi = \pi^G_{max}$.

Lemma 3. *Given a bargaining game G, for any deal $D \in \Omega(G)$,*

$$D \in \gamma(G) \text{ iff } \Phi_1 \subseteq D_1 \text{ and } \Phi_2 \subseteq D_2.$$

where (Φ_1, Φ_2) is the core of G.

Proof. "$\Rightarrow$" Straightforward from the definition of $\gamma(G)$.

"$\Leftarrow$" For any deal $D \in \Omega(G)$, if $\Phi_1 \subseteq D_1$ and $\Phi_2 \subseteq D_2$, then for each i, $X_i^{\leq \pi^G_{max}} \subseteq D_i$. It follows that $\max\{k : X_1^{\leq k} \subseteq D_1$ and $X_2^{\leq k} \subseteq D_2\} \geq \pi^G_{max}$. Therefore $\max\{k : X_1^{\leq k} \subseteq D_1$ and $X_2^{\leq k} \subseteq D_2\} = \pi^G_{max}$.

The above results show an intuitive procedure to construct a bargaining solution. First calculate the core by going through both parties's hierarchies of demands in parallel top-down to the level at which the collective demands are maximally consistent with the common demands. Then collect all the deals that contain the core. Finally, calculate the intersection of the deals that contain the core for each party.

Assume that X_1 and X_2 are two belief sets (so logically closed), representing the belief states of two agents. Mutual belief revision between the agents means that each agent takes part of the other agent's beliefs to revise his belief set. For

instance, if Ψ_1 is a subset of X_1 and Ψ_2 is a subset of X_2, then $X_1 \otimes_1 \Psi_2$ is the revised belief set of player 1 after he accepts player 2's beliefs Ψ_2 while $X_2 \otimes_2 \Psi_1$ is the resulting belief set of player 2 after accepting Ψ_1. Such an interaction of belief revision can continue until it reaches a fixed point where the beliefs in common, $(X_1 \otimes_1 \Psi_2) \cap (X_2 \otimes_2 \Psi_1)$, are exactly the beliefs that the agents mutually accept, $\Psi_1 + \Psi_2$. This gives

$$\Psi_1 + \Psi_2 = (X_1 \otimes_1 \Psi_2) \cap (X_2 \otimes_2 \Psi_1) \tag{4}$$

Suppose that the belief sets, X_1 and X_2, represent the two agents' demands, respectively. Then $(X_1 \otimes_1 \Psi_2) \cap (X_2 \otimes_2 \Psi_1)$ should represent the common revised demands after negotiation if Ψ_1 and Ψ_2 are the agreements that are mutually accepted each other. Therefore any bargaining solution should satisfy the fixed-point condition (4). The following theorem confirms that the solution we constructed in this paper satisfies the fixed-point condition.

Theorem 1. *For any bargaining game $G = ((X_1, \succeq_1), (X_2, \succeq_2))$, if X_1 and X_2 are logically closed, the bargaining solution $F(G)$ satisfies the following fixed-point condition:*

$$F_1(G) + F_2(G) = (X_1 \otimes_1 F_2(G)) \cap (X_2 \otimes_2 F_1(G)) \tag{5}$$

where $\otimes_i$ is the prioritized revision operator for player i.

To show this theorem, we need a few technical lemmas.

Lemma 4. *For any bargaining game $G = ((X_1, \succeq_1), (X_2, \succeq_2))$,*

1. *$F_1(G) \subseteq X_1 \otimes_1 F_2(G)$;*
2. *$F_2(G) \subseteq X_2 \otimes_2 F_1(G)$.*

Proof. According to the definition of prioritized base revision, we have $X_1 \otimes_1 F_2(G) = \bigcap_{H \in X_1 \Downarrow F_2(G)} Cn(H \cup F_2(G))$. For any $H \in X_1 \Downarrow F_2(G)$, there is a deal $(D_1, D_2) \in \Omega(G)$ such that $D_1 = H$. This is because we can extend the pair $(H, F_2(G))$ to a deal (H, D_2) such that $F_2(G) \subseteq D_2$. On the other hand, since $\Phi_1 \cup F_2(G)$ is consistent, we have $\Phi_1 \subseteq H$, where (Φ_1, Φ_2) is the core of G. Thus, $\Phi_1 \subseteq D_1$ and $\Phi_2 \subseteq D_2$. According to Lemma 3, we have $(D_1, D_2) \in \gamma(G)$. Since $F_1(G) \subseteq D_1$, we have $F_1(G) \subseteq H$. We conclude that $F_1(G) \subseteq X_1 \otimes_1 F_2(G)$. The proof of the second statement is similar.

By this lemma we have,

1. $F_1(G) + F_2(G) \subseteq X_1 \otimes_1 F_2(G)$;
2. $F_1(G) + F_2(G) \subseteq X_2 \otimes_2 F_1(G)$.

Note that the above lemma does not require the demand sets X_1 and X_2 to be logically closed. However, the following lemmas do.

Lemma 5. *Let (Φ_1, Φ_2) be the core of game $G = ((X_1, \succeq_1), (X_2, \succeq_2))$. If X_1 and X_2 are logically closed, then*

1. $X_1 \otimes_1 F_2(G) = X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2))$;
2. $X_2 \otimes_2 F_1(G) = X_2 \otimes_2 (\Phi_1 + (X_1 \cap X_2))$

Proof. We only present the proof of the first statement. The second one is similar. Firstly, we prove that $F_2(G) \subseteq \Phi_1 + \Phi_2 + (X_1 \cap X_2)$. If $X_1 \cup X_2$ is consistent, the result is obviously true. Therefore we can assume that $X_1 \cup X_2$ is inconsistent.

Assume that $\varphi \in F_2(G)$. If $\varphi \notin \Phi_1 + \Phi_2 + (X_1 \cap X_2)$, we have $\{\neg\varphi\} \cup \Phi_1 \cup \Phi_2 \cup (X_1 \cap X_2)$ is consistent. According to Lemma 2, we have $X_1^{\leq \pi^G_{max}+1} \cup X_2^{\leq \pi^G_{max}+1} \cup (X_1 \cap X_2)$ is inconsistent. Since our language is finite and both X_1 and X_2 are logically closed, the sets $X_1 \cap X_2$, $X_1^{\leq \pi^G_{max}+1}$ and $X_2^{\leq \pi^G_{max}+1}$ are all logically closed (the latter two due to LC). Therefore each set has a finite axiomatization. Let sentence ψ_0 axiomatize $X_1 \cap X_2$, ψ_1 axiomatize $X_1^{\leq \pi^G_{max}+1}$ and ψ_2 axiomatize $X_2^{\leq \pi^G_{max}+1}$. Thus $\psi_0 \wedge \psi_1 \wedge \psi_2$ is inconsistent. Notice that $\psi_0 \wedge \psi_1 \in X_1$ and $\psi_0 \wedge \psi_2 \in X_2$. It follows that $\neg\varphi \vee (\psi_0 \wedge \psi_1) \in X_1$ and $\neg\varphi \vee (\psi_0 \wedge \psi_2) \in X_2$. Since $\{\neg\varphi\} \cup \Phi_1 \cup \Phi_2 \cup (X_1 \cap X_2)$ is consistent, there is a deal $(D_1, D_2) \in \gamma(G)$ such that $\{\neg\varphi \vee (\psi_0 \wedge \psi_1)\} \cup \Phi_1 \cup (X_1 \cap X_2) \subseteq D_1$ and $\{\neg\varphi \vee (\psi_0 \wedge \psi_2)\} \cup \Phi_2 \cup (X_1 \cap X_2) \subseteq D_2$. We know that $\varphi \in F_2(G)$, so $\varphi \in D_1 + D_2$. Thus $\psi_0 \wedge \psi_1 \wedge \psi_2 \in D_1 + D_2$, which contradicts the fact that $D_1 + D_2$ is consistent. Therefore, we have shown that $F_2(G) \subseteq \Phi_1 + \Phi_2 + (X_1 \cap X_2)$.

Now we prove that $X_1 \otimes_1 F_2(G) = X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2))$. By Lemma 4, we have $\Phi_1 + \Phi_2 \subseteq X_1 \otimes_1 F_2(G)$. It follows that $X_1 \otimes_1 F_2(G) = (X_1 \otimes_1 F_2(G)) + (\Phi_1 + \Phi_2)$. Furthermore, we yield $X_1 \otimes_1 F_2(G) = (X_1 \otimes_1 F_2(G)) + (\Phi_1 + \Phi_2) + (X_1 \cap X_2)$ because $X_1 \cap X_2 \subseteq F_2(G)$. Since $F_2(G) \subseteq \Phi_1 + \Phi_2 + (X_1 \cap X_2)$. According to the AGM postulates, we have $(X_1 \otimes_1 F_2(G)) + (\Phi_1 + \Phi_2 + (X_1 \cap X_2)) = X_1 \otimes_1 (\Phi_1 + \Phi_2 + (X_1 \cap X_2))$. Therefore $X_1 \otimes_2 F_2(G) = X_1 \otimes_1 (\Phi_1 + \Phi_2 + (X_1 \cap X_2))$. In addition, it is easy to prove that $\Phi_1 \subseteq X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2))$. By the AGM postulates again, we have $X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2)) = (X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2)) + \Phi_1 = X_1 \otimes_1 (\Phi_1 + \Phi_2 + (X_1 \cap X_2))$. Therefore $X_1 \otimes_1 F_2(G) = X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2))$.

The following lemma will complete the proof of Theorem 1.

Lemma 6. *If X_1 and X_2 are logically closed, then*

$$(X_1 \otimes_1 F_2(G)) \cap (X_2 \otimes_2 F_1(G)) \subseteq F_1(G) + F_2(G).$$

Proof. et

$\Phi'_1 = X_1^{\leq \pi^1_{max}}$, where $\pi^1_{max} = \max\{k : X_1^{\leq k} \cup \Phi_2 \cup (X_1 \cap X_2)$ is consistent$\}$ and

$\Phi'_2 = X_2^{\leq \pi^2_{max}}$, where $\pi^2_{max} = \max\{k : \Phi_1 \cup X_2^{\leq k} \cup (X_1 \cap X_2)$ is consistent$\}$,

where (Φ_1, Φ_2) is the core of G.

Note that in the cases when π^i_{max} does not exist, we simply assume that it equals to $+\infty$. We claim that $X_1 \otimes_1 F_2(G) = \Phi'_1 + F_2(G)$ and $X_2 \otimes_2 F_1(G) = \Phi'_2 + F_1(G)$. We shall provide the proof of the first statement. The second one is similar.

Firstly, according to Lemma 2, $\Phi_1 \subseteq \Phi_1'$. Secondly, by Lemma 5, we have $X_1 \otimes_1 F_2(G) = X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2))$. Therefore to show $X_1 \otimes_1 F_2(G) = \Phi_1' + F_2(G)$, we only need to prove that $X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2)) = \Phi_1' + \Phi_2 + (X_1 \cap X_2)$. This is because $\Phi_2 + (X_1 \cap X_2) \subseteq F_2(G)$, $F_2(G) \subseteq \Phi_1 + \Phi_2 + (X_1 \cap X_2)$ and $\Phi_1 \subseteq \Phi_1'$. By the construction of prioritized revision, we can easily verify that $\Phi_1' + \Phi_2 + (X_1 \cap X_2) \subseteq X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2))$. Therefore we only have to show the other direction, i.e., $X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2)) \subseteq \Phi_1' + \Phi_2 + (X_1 \cap X_2)$.

If $\Phi_1' = X_1$, then $X_1 \cup (\Phi_2 + (X_1 \cap X_2))$ is consistent. It follows that $X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2)) \subseteq X_1 + (\Phi_2 + (X_1 \cap X_2)) = \Phi_1' + \Phi_2 + (X_1 \cap X_2)$, as desired. If $\Phi_1' \neq X_1$, according to the definition of π^1_{max}, we have $X_1^{\leq \pi^1_{max}+1} \cup \Phi_2 \cup (X_1 \cap X_2)$ is inconsistent. Therefore there exists $\psi \in X_1^{\leq \pi^1_{max}+1}$ such that $\neg\psi \in \Phi_2 + (X_1 \cap X_2)$. Now we assume that $\varphi \in X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2))$. If $\varphi \notin \Phi_1' + \Phi_2 + (X_1 \cap X_2)$, then $\{\neg\varphi\} \cup \Phi_1' \cup \Phi_2 \cup (X_1 \cap X_2)$ is consistent. So is $\{\neg\varphi \vee \psi\} \cup \Phi_1' \cup \Phi_2 \cup (X_1 \cap X_2)$. Notice that $\neg\varphi \vee \psi \in X_1^{\leq \pi^1_{max}+1}$. There exists $H \in X_1 \Downarrow (\Phi_2 + (X_1 \cap X_2))$ such that $\{\neg\varphi \vee \psi\} \cup \Phi_1' \subseteq H$. Since $\varphi \in X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2))$ and H is logically closed, we have $\psi \in H$, which contradicts the consistency of $H \cup (\Phi_2 + (X_1 \cap X_2))$. Therefore $X_1 \otimes_1 (\Phi_2 + (X_1 \cap X_2)) \subseteq \Phi_1' + \Phi_2 + (X_1 \cap X_2)$.

Finally we prove the claim of the lemma. Let $\varphi \in (X_1 \otimes_1 F_2(G)) \cap (X_2 \otimes_2 F_1(G))$. We then have $\varphi \in (\Phi_1' + F_2(G)) \cap (\Phi_1' + F_2(G))$. For $\varphi \in \Phi_1' + F_2(G)$, there exists a sentence ψ_2 such that $F_2(G) \vdash \psi_2$ and $\varphi \vee \neg\psi_2 \in \Phi_1'$. Similarly, there exists a sentence ψ_1 such that $F_1(G) \vdash \psi_1$ and $\varphi \vee \neg\psi_1 \in \Phi_2'$. It turns out that $\varphi \vee \neg\psi_1 \vee \neg\psi_2 \in \Phi_1' \cap \Phi_2'$. Thus $\varphi \vee \neg\psi_1 \vee \neg\psi_2 \in X_1 \cap X_2$. However, $X_1 \cap X_2 \subseteq F_1(G) + F_2(G)$. It follows that $\varphi \vee \neg\psi_1 \vee \neg\psi_2 \in F_1(G) + F_2(G)$. Note that $\psi_1 \wedge \psi_2 \in F_1(G) + F_2(G)$. Therefore we conclude that $\varphi \in F_1(G) + F_2(G)$.

4 Conclusion and Related Work

We have presented a logic-based bargaining solution based on Zhang and Zhang's model [12]. We have shown that the solution satisfies the fixed-point property, which asserts that the procedure of negotiation can be viewed as a course of mutual belief revision. The result is interesting not only because the result itself presents a desirable logical property of bargaining solutions but also establishes a link between bargaining and multi-agent belief revision. On the one hand, efforts have been made to the investigation of multi-agent belief revision [15,16], the research is far from satisfaction. On the other hand, bargaining have been a research topic in game theory for a few decades with sophisticated theory and variety of applications. It is easy to see that all the concepts introduced in this paper for the two-player bargaining game can be easily extended to the n-player cases. However, the extension of fixed-point property of mutual belief revision can be extremely hard. Therefore the link between bargaining and belief revision could give us a better understanding of multi-agent belief revision and could give us some hints towards the research.

The fixed-point property for negotiation functions was proposed by Zhang et al. [11]. However, there was no concrete negotiation function is constructed to satisfy the property. Meyer *et al.* gave a construction of negotiation function based on belief revision and discussed their logical properties [9,10]. Zhang and Zhang presented another belief-revision-based bargaining solution [12,13], which is similar to ours. However it is not too hard to verify that none of the above mentioned solutions satisfies the fixed-point property. Jin *et al.* [17] presents a mutual belief revision function that satisfies a fixed-point condition. However, the construction of the function is defined on belief revision operator and the fixed-point condition describes totally different property, which says that mutual belief revision is closed under iteration.

References

1. Nash, J.: The bargaining problem. Econometrica 18(2), 155–162 (1950)
2. Osborne, M.J., Rubinstein, A.: Bargaining and Markets. Academic Press, London (1990)
3. Binmore, K., Osborne, M.J., Rubinstein, A.: Noncooperative models of bargaining. In: Aumann, R., Hart, S. (eds.) Handbook of Game Theory with Economic Applications, vol. 1, pp. 180–225. Elsevier, Amsterdam (1992)
4. Thomson, W.: Cooperative models of bargaining. In: Aumann, R., Hart, S. (eds.) Handbook of Game Theory, vol. 2, pp. 1237–1284. Elsevier, Amsterdam (1994)
5. Rubinstein, A.: Economics and Language: Five Essays. Cambridge University Press, Cambridge (2000)
6. Kraus, S., Sycara, K., Evenchik, A.: Reaching agreements through argumentation: a logical model and implementation. Artificial Intelligence 104, 1–69 (1998)
7. Parsons, S., Sierra, C., Jennings, N.R.: Agents that reason and negotiate by arguing. Journal of Logic and Computation 8(3), 261–292 (1998)
8. Zhang, D., Zhao, K., Liang, C.M., Begum, G., Huang, T.H.: Strategic trading agents via market modeling. ACM SIGecom Exchanges, Special Issue on Trading Agent Design and Analysis 4(3), 46–55 (2004)
9. Meyer, T., Foo, N., Kwok, R., Zhang, D.: Logical foundations of negotiation: strategies and preferences. In: Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004), pp. 311–318 (2004)
10. Meyer, T., Foo, N., Kwok, R., Zhang, D.: Logical foundations of negotiation: outcome, concession and adaptation. In: Proceedings of the 19th National Conference on Artificial Intelligence (AAAI 2004), pp. 293–298 (2004)
11. Zhang, D., Foo, N., Meyer, T., Kwok, R.: Negotiation as mutual belief revision. In: Proceedings of the 19th National Conference on Artificial Intelligence (AAAI 2004), pp. 317–322 (2004)
12. Zhang, D., Zhang, Y.: A computational model of logic-based negotiation. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006), pp. 728–733 (2006)
13. Zhang, D., Zhang, Y.: Logical properties of belief-revision-based bargaining solution. In: Sattar, A., Kang, B.-h. (eds.) AI 2006. LNCS (LNAI), vol. 4304, pp. 79–89. Springer, Heidelberg (2006)
14. Nebel, B.: Syntax-based approaches to belief revision. In: Gärdenfors (ed.) Belief Revision, pp. 52–88. Cambridge University Press, Cambridge (1992)

15. Kfir-Dahav, N.E., Tennenholtz, M.: Multi-agent belief revision. In: Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge, pp. 175–194. Morgan Kaufmann Publishers Inc., San Francisco (1996)
16. Liu, W., Williams, M.A.: A Framework for Multi-Agent Belief Revision, Part I: The Role of Ontology. In: Foo, N.Y. (ed.) Canadian AI 1999. LNCS, vol. 1747, pp. 168–179. Springer, Heidelberg (1999)
17. Jin, Y., Thielscher, M., Zhang, D.: Mutual belief revision: semantics and computation. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI 2007), pp. 440–445 (2007)

Re-representation in a Logic-Based Model for Analogy Making

Ulf Krumnack, Helmar Gust, Kai-Uwe Kühnberger, and Angela Schwering

Institute of Cognitive Science, Osnabrück
{krumnack,hgust,kkuehnbe,aschweri}@uos.de

Abstract. Analogical reasoning plays an important role for cognitively demanding tasks. A major challenge in computing analogies concerns the problem of adapting the representation of the domains in a way that the analogous structures become obvious, i.e. finding and, in certain circumstances, generating appropriate representations that allow for computing an analogical relation. We propose to resolve this re-representation problem of analogy making in a logical framework based on the anti-unification of logical theories. The approach is exemplified using examples from qualitative reasoning (naive physics) and mathematics.

1 Introduction

The ability of analogy making is considered to be an essential part of intelligent behavior. By mapping knowledge about a well-known domain into a less familiar target domain, new hypotheses about that domain can be inferred. Moreover, the discovery of common structures can initiate a generalization process and support the introduction of abstract concepts, which are not directly observable.

Analogy making is based on a mapping of objects and relations between two domains. The creation of such an analogical mapping is well examined, provided the corresponding representations of both domains are already chosen in a way that the structural compatibility is obvious. Unfortunately, the structural commonalities characterizing two analogous domains are usually not obvious in advance, but become visible as a result of the analogy making process. The conceptualization must be adapted to make implicit analogous structures explicit. It is argued that a crucial part of establishing an analogy is a change of representation of one or both domains, the so-called re-representation [1]. In this paper, we propose a framework to deal with the problem of re-representation in a logic-based model for analogy making.

A variety of formal models for analogy making have been proposed [2] that employ very different representation schemes ranging from symbolic through hybrid architectures to connectionist systems. Therefore the notion of re-representation varies between the different approaches: Indurkhya [1] develops a theory in which the computation of analogies is based on the *accommodation* of an internal concept network to an input (resulting in a re-representation of the concept network), or the *projection* of a concept network to the input (resulting in a re-representation of the input), or both. In the Copycat model [3] representations

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 42–48, 2008.

for the source and target domain are constructed dynamically and in parallel. Although only little attention was paid to re-representation in the structure-mapping tradition at the beginning, in [4], a theory of re-representation in the SME tradition is presented which allows a restructuring based on operations such as transformation, decomposition, entity splitting etc. In that approach, mapping and re-representation are clearly separated processes. The DUAL/AMBR architecture [5] consists of a memory model in which knowledge is stored in a network of micro-agents. The same situation can evoke, depending on the context, different activation patterns which can be seen as a kind of re-representation.

2 A Logic-Based Model for Analogy Making

There exist different proposals to use anti-unification as a means to compute analogies [6,7,8]. The notion of anti-unification is based on the instantiation ordering of terms: given a first order language $\mathcal{L}$, a term t_2 is called an instance of another term t_1 (and vice versa t_1 is called an anti-instance of t_2), if there exists a substitution σ such that $t_1\sigma = t_2$. In this case, we write $t_1 \xrightarrow{\sigma} t_2$ or just $t_1 \rightarrow t_2$. The instantiation relation induces a preorder on the set of all terms.

Given a pair of terms s and t, an anti-unifier of s and t is a term g such that $s \leftarrow g \rightarrow t$. An anti-unifier g is called a least general anti-unifier, if g is an instance of every other anti-unifier. It has been shown that a least general anti-unifier exists for every pair of terms and that this anti-unifier is unique up to renaming of variables [9]. A natural idea is to extend the anti-unification of terms to the anti-unification of formulas [8,10].

The well-known Rutherford analogy is used for exemplification: Table 1 shows an axiomatization of the two domains for which the analogy shall be established. The source domain represents the knowledge about the solar system, stating that the mass of the sun is greater than the mass of a planet, that there is gravitation between the sun and the planet, and that for every pair of objects with gravitation between them, the lighter one will revolve around the heavier one provided a positive distance between the objects is conserved. On the target side we just know that the lightweight electrons are attracted by the nucleus due to coulomb force and that, despite of this attraction, atoms do not collapse. The latter fact, namely that electrons and nucleus have a distance greater than 0 is a formulation of the gold foil experiment due to Rutherford. Now anti-unification can be used to relate these two situations: for example, anti-unifying the two axioms α_1 and β_1 in Table 1 results in a generalized term γ_1 and two substitutions σ and τ such that it holds:

$$\text{mass(sun)} > \text{mass(planet)} \xleftarrow{\sigma = \{A \mapsto sun,\ B \mapsto planet\}} \text{mass}(A) > \text{mass}(B) \xrightarrow{\tau = \{A \mapsto nucleus,\ B \mapsto electron\}} \text{mass(nucleus)} > \text{mass(electron)}$$

By combining the two substitutions, an analogical relation between the domains can be established: *sun* corresponds to *nucleus* and *planet* corresponds to *electron*. This mapping is further supported by the fact, that also the axioms α_2

Table 1. A formalization of the Rutherford analogy with obvious analogous structures

Solar System	Rutherford Atom
$\alpha_1 : mass(sun) > mass(planet)$	$\beta_1 : mass(nucleus) > mass(electron)$
$\alpha_2 : \forall t : distance(sun, planet, t) > 0$	$\beta_2 : \forall t : distance(nucleus, electron, t) > 0$
$\alpha_3 : gravity(sun, planet) > 0$	$\beta_3 : coulomb(nucleus, electron) > 0$
$\alpha_4 : \forall x \forall y : gravity(x, y) > 0$ $\rightarrow attracts(x, y)$	$\beta_4 : \forall x \forall y : coulomb(x, y) > 0$ $\rightarrow attracts(x, y)$
$\alpha_5 : \forall x \forall y \forall t : attracts(x, y) \wedge$ $distance(x, y, t) > 0 \wedge$ $mass(x) > mass(y)$ $\rightarrow revolves\_arround(y, x)$	
Generalized Theory	
$\gamma_1 : mass(A) > mass(B)$	
$\gamma_2 : \forall t : distance(A, B, t) > 0$	
$\gamma_3 : F(A, B) > 0$	
$\gamma_4 : \forall x \forall y : F(x, y) > 0 \rightarrow attracts(x, y)$	

and β_2 can be anti-unified using the same substitutions. By slightly extending the notion of substitution and allowing the introduction of variables for function and predicate names, also the axioms α_3 and β_3 can be matched resulting in a mapping associating *gravity* and *coulomb*, which can be further supported by anti-unifying α_4 and β_4 with the same substitutions. Using this analogical relation, one can try to transfer the remaining, unmatched formulas from the source to the target. By transferring α_5 to the target one can infer that the electron revolves around the nucleus, i.e. the main claim of Rutherford's atom model.

Table 2 depicts a different axiomatization of the domains. Although all formulas from Table 1 still can be derived from Table 2, the generalized formula γ_3 cannot be discovered by anti-unifying the given formulas. Notice further, that the parameters of β'_2 are switched compared to β_2, so that anti-unifying α'_2 and β'_2 would result in in the unwanted mapping of *sun* to *electron* and *planet* to *nucleus*, which contradicts the mapping established by anti-unifying α'_1 and β'_1.

In this situation, a new representation of the given axiomatization is needed, that exhibits the common structure of the two domains in a way that anti-unification leads to an appropriate generalized theory. For example, from the background knowledge it is known that *distance* is a symmetric function, i.e. $distance(nucleus, electron, t) > 0$ can be derived from $\{\phi_1, \beta'_2\}$, which is a good candidate for anti-unification with α'_3. Similarly, $gravity(sun, planet) > 0$ can be derived from $\{\phi_2, \alpha'_1, \alpha'_2, \alpha'_4\}$ on the source side, which can be anti-unified with $coulomb(nucleus, electron) > 0$, which can be inferred from $\{\beta'_3, \beta'_4, \beta'_5\}$ on the target side. Hence, the task of re-representation consists of finding pairs of formulas from the domain theories, that possess a common structure.

While in this example, re-representation is just needed to enhance the support for the analogy, there are cases, where no analogy can be computed at all, if the given representation for two domains is not altered. Here the choice of logic as a representation formalism exhibits its power, since beside the formulas explicitly given, there are also implicit formulas that can be inferred from these axioms. This provides a natural notion of re-representation, where other approaches for analogy making have to introduce special and sometimes quite artificial means.

Table 2. A formalization of the Rutherford analogy where re-representation is required

Background Knowledge	
$\phi_1 : \forall x \forall y \forall t : distance(x, y, t) = distance(y, x, t)$	
$\phi_2 : \forall x \forall y \forall z : x > y \wedge y > z \rightarrow x > z$	
Solar System	Rutherford Atom
$\alpha'_1 : mass(sun) > mass(planet)$	$\beta'_1 : mass(nucleus) > mass(electron)$
$\alpha'_2 : mass(planet) > 0$	$\beta'_2 : \forall t : distance(electron, nucleus, t) > 0$
$\alpha'_3 : \forall t : distance(sun, planet, t) > 0$	$\beta'_3 : charge(nucleus) > 0$
$\alpha'_4 : \forall x \forall y : mass(x) > 0 \wedge mass(y) > 0$	$\beta'_4 : charge(electron) < 0$
$\rightarrow gravity(x, y) > 0$	$\beta'_5 : \forall x \forall y : charge(x) > 0 \wedge charge(y) < 0$
$\alpha'_5 : \forall x \forall y : gravity(x, y) > 0$	$\rightarrow coulomb(x, y) > 0$
$\rightarrow attracts(x, y)$	$\beta'_6 : \forall x \forall y : coulomb(x, y) > 0$
$\alpha'_6 : \forall x \forall y \forall t : attracts(x, y) \wedge$	$\rightarrow attracts(x, y)$
$distance(x, y, t) > 0 \wedge$	
$mass(x) > mass(y)$	
$\rightarrow revolves\_arround(y, x)$	

3 Formal Treatment

3.1 Anti-unification of Theories

For a set of formulas F, we will denote the theory of F, i.e. the set of all formulas that can be inferred from F, by $Th(F)$. We will call a set of formulas F_1 an *anti-instance* of a set F_2, if there exists a substitution σ such that $Th(F_1\sigma) \subseteq Th(F_2)$.[1] In this case we write $F_1 \xrightarrow{\sigma} F_2$ or just $F_1 \rightarrow F_2$. If $Th(F_1\sigma) = Th(F_2)$, F_1 is called a *full anti-instance* of F_2 and we write $F_1 \overset{\sigma}{\Rightarrow} F_2$ or just $F_1 \Rightarrow F_2$.

Given two sets of formulas Ax_S and Ax_T, we will call a triple $\langle G, \sigma, \tau \rangle$, consisting of a finite set of formulas G and substitutions σ and τ, an *anti-unifier* of Ax_S and Ax_T, iff $Ax_S \xleftarrow{\sigma} G \xrightarrow{\tau} Ax_T$. An anti-unifier $\langle G, \sigma, \tau \rangle$ is *at least as specific* as $\langle G', \sigma', \tau' \rangle$, if G' is a full anti-instance of G in a way that is compatible with the domain substitutions, i.e. if $G' \overset{\theta}{\Rightarrow} G$ then $\sigma \circ \theta = \sigma'$ and $\tau \circ \theta = \tau'$. Using a most specific anti-unifier can help to prevent proliferation of variables as indicated by the following example: consider the sets $Ax_S = \{p(a), q(a)\}$ and $Ax_T = \{p(b), q(b)\}$. Then $G' = \{p(X), q(Y)\}$ with substitutions $\sigma' = \{X \mapsto a, Y \mapsto a\}$ and $\tau' = \{X \mapsto b, Y \mapsto b\}$ is an anti-unifier. Notice, that G' consists only of least general anti-unifiers of formulas from Ax_S and Ax_T, but as a set of formulas it is not least general by itself, since $G = \{p(X), q(X)\}$ and the substitutions $\sigma = \{X \mapsto a\}, \tau = \{X \mapsto b\}$ fulfill $G = G'\{Y \mapsto X\}, \sigma' = \sigma \circ \{Y \mapsto X\}, \tau' = \tau \circ \{Y \mapsto X\}$. Therefore in the context of analogy making we will only apply least general anti-unifiers, since any more general anti-unifier only adds complexity without extending the analogical relation.

3.2 Coverage

Obviously, least general anti-unifiers for sets of formulas always exist: the minimal example is the empty set $\emptyset$. This is probably not a desirable solution, since it results in the empty analogical relation. Therefore we introduce the concept

[1] We consider only admissible substitution, i.e. substitutions that don't introduce variables into the scope of a quantifier.

Table 3. Addition and multiplication of natural numbers

Addition	Multiplication
$\alpha_1 : \forall x : add(0, x) = x$	$\beta_1 : \forall x : mul(0, x) = 0$
$\alpha_2 : \forall x \forall y : add(s(y), x) = add(y, s(x))$	$\beta_2 : \forall x \forall y : mul(s(y), x) = add(mul(y, x), x)$

of coverage: given an anti-unifier $\langle G, \sigma, \tau \rangle$ for Ax_S and Ax_T, the subset $Th(G\sigma)$ of $Th(Ax_S)$ is said to be covered by G and for Ax_T analogously.

An anti-unifier $\langle G, \sigma, \tau \rangle$ has a greater or equal coverage than $\langle G', \sigma', \tau' \rangle$ if there is a substitution $G' \xrightarrow{\theta} G$ that is compatible with the domain substitutions. In general, a greater coverage is preferable, since it provides more support for the analogy. However, there are some caveats. Assume for example the axioms for addition and multiplication of natural numbers given in Table 3. There is no direct anti-unifier for the axioms, but since we can derive[2] $\beta_3 : \forall x : mul(s(0), x) = x$ we get an anti-unifier $\langle \{\gamma_1\}, \sigma, \tau \rangle$ with $\gamma_1 : \forall x : \mathbf{Op}(\mathbf{E}, x) = x$ and substitutions $\sigma = \{\mathbf{Op} \mapsto add, \mathbf{E} \mapsto 0\}$ and $\tau = \{\mathbf{Op} \mapsto mul, \mathbf{E} \mapsto s(0)\}$ which expresses the intended analogy, i.e. that addition corresponds to multiplication and that there is a unit element, 0 for addition and $s(0)$ for multiplication. Notice, that only α_1 is covered by this anti-unifier. We can extend the coverage of the anti-unifier by adding formulas, e.g. an arbitrary number of formulas of the type[3]

$$\mathbf{Op}(s^n(0), s^m(0)) = \mathbf{Op}(s^m(0), s^n(0))$$

This can be done without changing the substitutions, hence every formula added will enhance the support for the analogy. But still α_1 is the only axiom covered. To address this problem, we need a concept of *indirect coverage*. We call those axioms indirectly covered which have been used to provide the formulas that are added. $\langle G, \sigma, \tau \rangle$ is a *full* anti-unifier, if all domain axioms are (indirectly) covered by it.

In total, to compute an analogical relation, the (indirect) coverage of the domains should be maximized while one-to-many mappings should be avoided.

4 Algorithm

Here we sketch a proposal for an algorithm to compute a generalized theory for given source and target domains. We assume that the axioms are in clause form.

A1 Sort the axioms of the target domain by a heuristic complexity measure, e.g. based on the number of literals per clause and the arity of the literals.

A2 Select the next least complex and yet uncovered axiom from the target domain. If no such axioms are left, then, if all axioms are covered, terminate, otherwise backtrack to step A3.

[2] Here we use $mul(s(0), x) = add(mul(0, x), x) = add(0, x) = x$, i.e. we use the axiom α_1 of the source theory to derive formulas on the target side!

[3] Notice that we cannot derive $\forall x \forall y : \mathbf{Op}(x, y) = \mathbf{Op}(y, x)$ in first order logic, and therefore all of these formulas are independent from each other.

A3 Find (non-deterministically) the best matching clauses from the source according to a heuristic distance measure. A simple distance measure uses the relative number of common functors and constants in corresponding subsets of the clauses. The current analogical relation should be used to identify unequal functors, for which the analogical relation already holds. If not possible, backtrack to step A2 (skip axiom).

A4 Split the clauses into matching literals and residual literals. The anti-unifier of the matching literals becomes a generalized clause, if the residual literals can be refuted.[4] Else, backtrack to step A3 (selecting a different clause).

A5 Check the cross domain consistency by projecting the residual literals to the other domain using the current analogical relation and try to prove them there. If this is possible, reject the match and backtrack to step A3 (selecting a different clause).

A6 Mark the clauses of the target domain, that are used in the refutation process (step A4) as (indirectly) covered. Repeat from step A2.

Using clause form for source and target axioms ensures that re-representation of the domains is a guided process. The compared clauses can only become smaller (by refuting the rest). This means that re-representation does not dramatically increase complexity additionally to the complexity of the proofs (which of course is undecidable as we allow full first-order logic). We demonstrate the algorithm using our second example of the Rutherford analogy (Table 2):

1. $\beta'_3 \leftrightarrow \alpha'_2$	select β'_3 from target (A2), find α'_2 as a best source match (A3) and anti-unify: $mass \leftrightarrow charge, planet \leftrightarrow nucleus$
2. $\beta'_4 : skipped$	no reasonable matching clause
3. $\beta'_2 \leftrightarrow \alpha'_3$	$sun \leftrightarrow electron$
4. $\beta'_1 : skipped$	$nucleus \neq electron, sun \neq nucleus$
5. $\beta'_6 \leftrightarrow \alpha'_5$	$gravity \leftrightarrow coulomb$
6. $\beta'_5 \leftrightarrow \alpha'_4$	refuting the residuals (A4) $\neg(charge(y) < 0)$ with $y = electron$ $\neg(mass(y) > 0)$ with $y \in \{sun, planet\}$ but the projection to the source of $\neg(charge(electron) < 0)$ namely $\neg(mass(sun) < 0)$ can be proven in the source (A5): backtrack to 1.
7. $\beta'_3 \leftrightarrow \psi_1$	$mass \leftrightarrow charge, sun \leftrightarrow nucleus$, where $\psi_i = mass(sun) > 0$, wich can be derived from $\alpha'_1, \alpha'_2, \phi_2$
...	like 2.-5.
12. $\beta'_5 \leftrightarrow \alpha'_4$	like 6.: backtrack to 1.
13. $\beta'_3 : skipped$	no other matching clause
14. $\beta'_4 : skipped$	no reasonable matching clause
15. $\beta'_2 \leftrightarrow \alpha'_3$	$sun \leftrightarrow electron, planet \leftrightarrow nucleus$
16. $\beta'_1 : skipped$	$nucleus \neq electron, sun \neq nucleus$ (avoid many-to-many mapping)
17. $\beta'_6 \leftrightarrow \alpha'_5$	$gravity \leftrightarrow coulomb$
18. $\beta'_5 \leftrightarrow \alpha'_4$	refuting the residuals $\neg(mass(x) > 0) \vee \neg(mass(y) > 0)$ $\neg(charge(x) > 0) \vee \neg(charge(y) < 0)$ solution, but β'_1 not covered: backtrack to 15.
19. $\beta'_2 \leftrightarrow \psi_2$	$sun \leftrightarrow nucleus, planet \leftrightarrow electron$ with $\psi_2 = distance(planet, sun, t) > 0$ derivable from α'_3, ϕ_1
20. $\beta'_1 \leftrightarrow \alpha'_1$	
21. $\beta'_6 \leftrightarrow \alpha'_5$	$gravity \leftrightarrow coulomb$
22. $\beta'_5 \leftrightarrow \alpha'_4$	refuting the residuals $\neg(mass(x) > 0) \vee \neg(mass(y) > 0)$ $\neg(charge(x) > 0) \vee \neg(charge(y) < 0)$ solution, target covered: termination

[4] For refutation a classical connection graph-based resolution prover can be used. During the refutation, variables may be instantiated. Therefore, anti-unification of the matching parts can only be done after these proofs.

This example demonstrates that there might exist different analogical mappings between source and target that employ substitutions with a different degree of complexity and that cover different parts of the domains. The process of constructing a preferred mapping is controlled by the combination of simple heuristics: a similarity measure for clauses is used to find a best matching candidate (A3) or to skip certain clauses without mapping (steps 2, 13, 14, 15). Logical inconsistencies can cause the algorithm to reject the first selections (A4 and A5) and backtrack with alternative candidates (steps 6, 12, 18).

5 Conclusion and Future Work

In this paper, it is shown how logical inference provides a way to tackle the problem of re-representations in analogical reasoning applications. This is essential for analogy making, as the assumption that the two domains are already available in a structure suitable for the analogical mapping is unrealistic. In fact, the analogous structure is a result of the analogy making process. An algorithm is proposed that computes the best matching clauses according to a heuristic distance measure. Besides a thorough practical evaluation of the proposed approach, future work concerns a theoretical assessment of the trade-off between the maximization of coverage and the minimization of substitution complexities.

References

1. Indurkhya, B.: Metaphor and cognition: an interactionist approach. Kluwer Academic Publishers, Dordrecht (1992)
2. Kokinov, B., French, R.M.: Computational models of analogy-making. In: Nadel, L. (ed.) Encyclopedia of Cognitive Science, vol. 1, pp. 113–118. Nature Publishing Group, London (2003)
3. Hofstadter, D. and The Fluid Analogies Research Group. Fluid concepts and creative analogies, New York (1995)
4. Yan, J., Forbus, K., Gentner, D.: A theory of rerepresentation in analogy matching. In: Proceedings of the Twenty-Fifth Annual Meeting of the Cognitive Science Society (2003)
5. Kokinov, B., Petrov, A.: Integration of memory and reasoning in analogy-making: The ambr model. In: Gentner, D., Holyoak, K., Kokinov, B. (eds.) The Analogical Mind: Perspectives from Cognitive Science. MIT Press, Cambridge (2001)
6. Hasker, R.W.: The replay of program derivations. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA (1995)
7. Burghardt, J.: E-generalization using grammars. Artificial Intelligence Journal 165(1), 1–35 (2005)
8. Gust, H., Kühnberger, K.U., Schmid, U.: Metaphors and heuristic-driven theory projection (HDTP). Theoretical Computer Science 354(1), 98–117 (2006)
9. Plotkin, G.D.: A note on inductive generalization. Machine Intelligence 5, 153–163 (1969)
10. Krumnack, U., Schwering, A., Gust, H., Kühnberger, K.U.: Restricted higher-order anti-unification for analogy making. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 273–282. Springer, Heidelberg (2007)

Knowledge Generation for Improving Simulations in UCT for General Game Playing

Shiven Sharma, Ziad Kobti, and Scott Goodwin

Department of Computer Science, University of Windsor,
401 Sunset Avenue, Windsor, ON N9C4B9, Canada
{sharmaw,kobti,sgoodwin}@uwindsor.ca
http://www.cs.uwindsor.ca

Abstract. General Game Playing (GGP) aims at developing game playing agents that are able to play a variety of games and, in the absence of pre-programmed game specific knowledge, become proficient players. Most GGP players have used standard tree-search techniques enhanced by automatic heuristic learning. The UCT algorithm, a simulation-based tree search, is a new approach and has been used successfully in GGP. However, it relies heavily on random simulations to assign values to unvisited nodes and selecting nodes for descending down a tree. This can lead to slower convergence times in UCT. In this paper, we discuss the generation and evolution of domain-independent knowledge using both state and move patterns. This is then used to guide the simulations in UCT. In order to test the improvements, we create matches between a player using standard the UCT algorithm and one using UCT enhanced with knowledge.

Keywords: General Game Playing, Monte Carlo Methods, Reinforcement Learning, UCT.

1 Introduction

Historically, game playing agents were designed to be good in specific games. However, even though these players excelled in games that they were designed for, they could not play any other games. General Game Playing (GGP) focuses on the creation of agents that are able to accept rules of a game, and use them to learn how to play it, eventually displaying a high level of competence in it.

A class of games for which a GGP approach was taken were *positional games* (eg. Tic-Tac-Toe, Hex and the Shannon switching games), which were formalised by [1]. A positional game can be defined by three sets, P, A, B. Set P is a set of positions; with set A and B are sets of subsets of P. Programs that are capable of accepting rules of positional games and, with practice, learn how to play the game have been developed. [1] constructed a program that is able to learn important board configurations in a 4 x 4 x 4 Tic-Tac-Toe game.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 49–55, 2008.

The annual General Game Playing Competition [2] organised by Stanford University has been instrumental in bringing about renewed interest in GGP. The rules of the games are written in Game Description Language (GDL) [3]. The tournaments are controlled by the Game Manager (GM) which relays the game information to each Game Player (GP) and checks for legality of moves and termination of the game [4]. Communication between players and the GM takes place in the form of HTTP messages. Successful players have mainly focused on automatically generating heuristics based on certain generic features identified in the game. Cluneplayer [5] was the winner of the first GGP competition, followed by Fluxplayer [6]. Both these players, along with UTexas Larg [7] use automatic feature extraction. Another approach that has been taken is in [8], where transfer of knowledge extracted from one game to another is explored by means of a TD(λ) based reinforcement learner.

The main aim of this paper is to explore the creation and use of domain-independent knowledge. The knowledge is then applied to UCT search for GGP. The rest of the paper is organised as follows. In the Section 2 the UCT search mechanism if discussed briefly along with its application to GGP. In Section 3 we discuss the knowledge representation scheme. Matches are then played between a standard UCT player and a player using the knowledge and stored tree for the games large (5 x 5) Tic-Tac-Toe, Connect-4, Breakthrough and Checkers.

2 UCT and General Game Playing

UCT [9] is an extension of the UCB1 algorithm [10], and stands for *Upper Confidence Bounds applied to Trees*. The UCB1 Algorithm aims at obtaining a fair balance between exploration and exploitation in a K-Armed bandit problem, in which the player is given the option of selecting one of K arms of a slot machine (i.e. the bandit). The selection of arms is directly proportional to the total number of plays and inversely proportional to the number of plays of each arm. This is seen in the selection formula

$$\bar{X_j} + C\sqrt{\frac{2\log n}{T_j(n)}} \tag{1}$$

The arm maximising (1) is selected. $\bar{X_j}$ is the average return of arm j after n plays, and $T_j(n)$ is the number of times it has been selected. C controls the balance between exploration and exploitation of the tree. This formula is used in our player.

UCT extends UCB1 to trees. A single simulation consists of selecting nodes to descend down the tree using and using (1) and random simulations to assign values to nodes that are being seen for the first time. CADIA-Player [11] was the first General Game Player to use a simulation based approach, using UCT to search for solutions, and was the winner of the last GGP Competition. UCT has also been used in the game of Go, and the current computer world champion, MoGo [12], uses UCT along with prior game knowledge. [13] also used simulations to

build a basic knowledge base of move sequence patterns in a multi-agent General Game Player. Selection of moves was done based on the average returns, and mutation between move sequence patterns was done to facilitate exploration. An advantage of a simulation based approach to General Game Playing is that for any game, generating a number of random game sequences does not consume a lot of time, since no effort is made in selecting a good move. UCT is able to guide these random playoffs and start delivering near-optimal moves. However, even with UCT, lack of game knowledge can be a significant obstacle in more complex games. This is because in the absence of any knowledge to guide the playoffs, it takes a large number of runs of the UCT algorithm to converge upon reasonably good moves.

2.1 Structure of the Game Player

In order to make a move, the current state is set as the root node of a tree which is used by UCT. To allow for adequate exploration, the algorithm ensures that each child is visited at least once. To manage memory, every node that is visited at least once is added to a *visited table*. Once a node is reached which is not in the table, a random simulation is carried on from that node till the end of the game. The reward received is backed up from that node to the root. Algorithm 1 shows the basic method of UCT and the update of values in a 2 player zero-sum game. A number of such simulations are carried out, each starting from the root and building the tree asymmetrically.

Algorithm 1. UCTSearch(*root*)

```
1: node = root
2: while visitedTable.contains(node) do
3:     if node is leaf then
4:         return value of node
5:     else
6:         if node.children.size == 0 then
7:             node.createChildren()
8:         end if
9:         selectedChild = UCTSelect(node)
10:        node = selectedChild
11:    end if
12: end while
13: visitedTable.add(node)
14: outcome = RandomSimulation(node)
15: while node.parent ≠ null do
16:    node.visits = node.visits + 1
17:    node.wins = node.wins + outcome
18:    outcome = 1 − outcome
19:    node = node.parent
20: end while
```

The value of each node is expressed as the ratio of the number of winning sequences through it to the total number of times it has been selected. In order to select the final move to be made, the algorithm returns the child of the root having the best value.

In the next section we discuss how domain-independent knowledge can be gathered from simulations to guide them towards reasonable solutions.

3 Knowledge Creation and Evolution

Knowledge can be used to guide node selection and random simulations in UCT. [12] uses domain dependent knowledge in the form of local patterns to make the simulations more realistic. [11] uses a history heuristic for moves to provide basic knowledge, associating with each move its average win rate, irrespective of the state it was made in. [14] use TD(0) learning to learn an evaluation function for a template of local features in Go. The value function is expressed as a linear combination of all local shape features, squashed by the *sigmoid* function to constrain it to be between $[0, 1]$, thus giving it a natural interpretation as a probability of winning. Our approach to learning state-space knowledge is very similar to it, the major difference being that instead of focusing on local features, we use the entire state description.

3.1 State-Space Knowledge

States in GDL are represented as a set of ordered tuples, each of which specifies a certain feature of the state. For example, in Tic-Tac-Toe, $cell(1, 1, b)$ specifies that the cell in row 1 and column 1 is blank. Therefore, a state in Tic-tac-Toe is represented as a set of 9 such tuples, each specifying whether a cell is blank or contains an X or an O. In order to extract state-space knowledge from such tuples, we assign values to each of these tuples. Initially, the values are set to 0, thereby giving the squashed state value as 0.5 (equal probability of winning and losing). The value of a state S is calculated as

$$V(S) = \sigma \left(\sum_{t \in match(S)} \psi_t \right) \tag{2}$$

where the sigmoid function $\sigma(t)$, a special form of the logistic function, squashes the value between 0 and 1. ψ_t is the value of tuple t that is part of the description of state S.

3.2 Move Knowledge

The manner in which move knowledge is expressed is similar to that used by [11]. However, instead of simply giving a move its average win rate as a value, we factor in the depth as well, as given in [15]. Assuming a move m is made at

position p during a random simulation, and the length of the simulation is l and the outcome is r, then the value assigned to m after n plays is given as

$$V(m) = \frac{\sum_{i=1}^{n} 2^{\frac{l_i - p_i}{l_i}} \times r_i}{n} \tag{3}$$

3.3 Learning

Every time a random game is played, knowledge is updated for the role(s) that used the knowledge (examples of roles are *black* and *white* in Chess). The result of the random game is used to perform the updates. If the outcome of game n is r, the values for each move played in the sequence, $V(m)$, is updated as

$$V(m) = \frac{R_{n-1} + 2^{\frac{l-p}{l}} \times r}{n} \tag{4}$$

where R_{n-1} is the cumulative reward for $n-1$ plays (the numerator of (3)).

For each tuples t in each state of the sequence, its value ψ_t is updated as

$$\psi_t = \psi_t + \alpha \left(\frac{r - V(S)}{size_S} \right) \tag{5}$$

The learning rate, α is gradually decreased over time. $size_S$ is the number of tuples in the description of S.

3.4 Using the Knowledge

Given the two forms of knowledge, $\Psi_m = V(m) + V(S_m)$ gives a measure of the effectiveness of a move, where m is the move which leads to state S_m. Random simulations use this knowledge by using $\epsilon - greedy$ selection, with a relatively high value of ϵ (this is done to prevent the simulations from becoming too deterministic).[1]

In order to select nodes that have not been selected previously (not in the *visitedTable* of Algorithm 1), UCT selects all nodes randomly, with equal probability. However, using the knowledge learnt it is possible to bias this selection in favour of nodes with higher values. Therefore, we assign these nodes an initial value based on $V(S_m)$ and $V(m)$.. This is similar to the approach taken in [16]. Node selection is then done using the standard UCT formula of (1), with $\bar{X}_j$ being replaced by Ψ_m. $N.value$ and $N.visits$ are updated as given in (1).

4 Experiments

In order to test the effectiveness of using knowledge with UCT, we played matches between a player using standard UCT, UCT_S, and a player using knowledge for simulations, UCT_K. We implemented a GGP type system on our local

[1] [16] presents an interesting discussion on the effect of different types of simulations in UCT search.

Table 1. Number of wins for UCT_S and UCT_K over 100 matches per game

Game	Wins for UCT_S	Wins for UCT_K	Total Draws
Tic-Tac-Toe (large)	7	15	78
Connect-4	41	59	0
Breakthrough	32	68	0
Checkers	43	57	0

machines. The players and the manager were written in Java. For inferencing, the game rules were converted to Prolog. YProlog [17], a Prolog-style inference engine written in Java, was used. The games used for testing were 5 X 5 Tic-Tac-Toe, Connect-4 and Breakthrough, Checkers. The results for 100 games each are shown. The knowledge, once generated, was not changed. The roles for the games were randomly distributed between the UCT_S and UCT_K. The addition of knowledge during random simulations results in corresponding player winning the majority of matches. We will investigate the effect continuous evolution of knowledge in future experiments. We also hope to reconstruct some of the existing game players (to the best of our knowledge) and test knowledge enhanced versions of the UCT player against them, as this will give a broader perspective as to the strength of the UCT player.

5 Conclusions and Future Work

In this paper we proposed an approach for generating and evolving domain-independent knowledge for General Game Players. The idea of the history heuristic was taken from [15], and it has been used in [11]. The state knowledge representation and learning was inspired from [14]. The combined knowledge is used to guide simulations in UCT. The results of matches between a standard UCT player and a player enhanced with knowledge clearly indicate the advantages of including knowledge. Learning can be speeded up by using faster inference engines (for example, YAP [18] and B-Prolog [19]), using hashing techniques such as [20] to look up tuples, and by parallelising the simulations.

The very nature of General Game Playing makes it difficult to create and use knowledge. Given the vast variety of games that can be played, building a general framework for knowledge representation and learning is challenging. In many games, moves that are successful in certain states may not be successful in other states. Therefore, the history value of moves only acts as a simple guide for making future moves. In our representation for state knowledge, we have assumed that the tuples are independent of each other. However, in most cases it is the relationship between various state features that matters. An alternative to using linear function approximation is using non-linear function approximation techniques such as neural-networks. We are also investigating the use of Ant Colony Optimisation approaches [21] to generate random sequences and use them in conjunction with the ideas presented in this paper.

Acknowledgments. This work was funded in part by an NSERC Discovery grant.

References

1. Koffman, E.: Learning through pattern recognition applied to a class of games. IEEE Trans. on Systems, Man and Cybernetics SSC-4 (1968)
2. Genesereth, M., Love, N.: General game playing: Overview of the aaai competition. AI Magazine, Spring 2005 (2005)
3. Genesereth, M., Love, N.: General game playing: Game description language specification, `http://games.standford.edu/competition/misc/aaai.pdf`
4. `http://games.stanford.edu`
5. Clune, J.: Heuristic evaluation functions for general game playing. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (2007)
6. Schiffel, S., Thielscher, M.: Fluxplayer: A successful general game player. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, pp. 1191–1196 (2007)
7. Banerjee, B., Kuhlmann, G., Stone, P.: Value function transfer for general game playing. In: ICML Workshop on Structural Knowledge Transfer for ML (2006)
8. Banerjee, B., Stone, P.: General game playing using knowledge transfer. In: The 20th International Joint Conference on Artificial Intelligence, pp. 672–777 (2007)
9. Kocsis, L., Szepesvari, C.: Bandit based monte-carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
10. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite time analysis of the multi-armed bandit problem. Machine Learning 47(2/3), 235–256 (2002)
11. Björnsson, Y., Finnsson, H.: Simulation-based approach to general game playing. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI Press, Menlo Park (2008)
12. Gelly, S., Wang, Y.: Modifications of uct and sequence-like simulations for monte-carlo go. In: IEEE Symposium on Computational Intelligence and Games, Honolulu, Hawaii (2007)
13. Sharma, S., Kobti, Z.: A multi-agent architecture for general game playing. In: IEEE Symposium on Computational Intelligence and Games, Honolulu, Hawaii (2007)
14. Silver, D., Sutton, R., Muller, M.: Reinforcement learning of local shape in the game of go. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007 (2007)
15. Schaeffer, J.: The history heuristic and alpha-beta search enhancements in practice. IEEE Transaction on Pattern Analysis and Machine Intelligence, 1203–1212 (1989)
16. Gelly, S.: A Contribution to Reinforcement Learning; Application to Computer-Go. PhD thesis, University of Paris South (2007)
17. Winikoff, M.: `http://www3.cs.utwente.nl/ schooten/yprolog`
18. `http://www.dcc.fc.up.pt/ vsc/Yap/`
19. `http://www.probp.com/`
20. Zobrist, A.: A new hashing method with application for game playing. Technical report 99, University of Wisconsin (1970)
21. Sharma, S., Kobti, Z., Goodwin, S.: General game playing with ants. In: The Seventh International Conference on Simulated Evolution And Learning (SEAL 2008), Melbourne, Australia (in press, 2008)

Propositional Automata and Cell Automata: Representational Frameworks for Discrete Dynamic Systems

Eric Schkufza, Nathaniel Love, and Michael Genesereth

Stanford University, CA, USA
eschkufz@cs.stanford.edu, nlove@cs.stanford.edu,
genesereth@cs.stanford.edu

Abstract. This paper describes and compares two simple, powerful models for formalizing the behavior of discrete dynamic systems: Propositional and Cell Automata. Propositional Automata encode state in terms of boolean propositions, and behavior in terms of boolean gates and latches. Cell Automata generalize the propositional model by encoding state in terms of multi-valued cells, and behavior in terms of comparators and selectors that respond to cell values. While the models are equally expressive, Cell Automata are computationally more efficient than Propositional Automata. Additionally, arbitrary Propositional Automata can be converted to optimal Cell Automata with identical behavioral properties, and Cell Automata can be encoded as a Propositional Automata with only logarithmic increase in size.

1 Introduction

Propositional Automata and Cell Automata are two models for formalizing the behavior of discrete dynamic systems. A Propositional Automaton is a circuit-based formalism—a network of boolean gates and latches that represent propositions, where inputs and outputs are represented as distinguished latches—analogous to the circuit-based formalisms for such systems described by Russell and Norvig [1]. A Cell Automaton is a compact, computationally-advantageous generalization of Propositional Automata: a network of multi-valued latches, comparators and selectors where inputs and outputs are represented as distinguished latches.

In this paper, we demonstrate how Cell Automata may contain logarithmically fewer structural elements and may require exponentially fewer operations for state update than equivalent Propositional Automata, and present an algorithmic transformation to take advantage of this efficiency gain. In some cases, however, a Propositional Automaton representation may reveal advantageous structural properties of a dynamic system. Thus this paper also presents a transformation from Cell to Propositional Automata resulting only in a logarithmic increase in the size of the representation.

Compared to existing formalisms for discrete dynamic systems, Propositional and Cell Automata offer several advantages. First, these automata are easy to understand: Cell Automata selectors resemble *if-then* structures of programming languages and the *rules* of Abstract State Machines (ASMs) [2]; their graphical structure analogizes to the

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 56–66, 2008.

serial and *flow* constructs of the business process language BPEL [3]; and propositions and multi-valued latches are both similar to Petri Net [4] *places*: objects held in cells of a current state determine the transition to the next state. By explicitly modeling inputs to dynamic systems, however, these automata distinguish themselves from ASMs, which are designed to model closed rather than open systems. Further, automata modeling separate systems may be composed by linking inputs and outputs, without modifying internal structures; this type of privacy-preserving composition cannot be performed with Petri Nets.

Second, Propositional and Cell Automata each have particular strengths for reasoning tasks, and as noted above, transformations between the formalisms are direct and can prove advantageous. Cell Automata can be evaluated in data-flow fashion in time linear in the number of latches that they contain. Thus, their compact formalization represents a computational advantage. Nonetheless, representing a system as a Propositional Automaton may reveal critical structural properties, including independent substructures, factors, bottlenecks, or symmetries.

Finally, the use of Propositional and Cell Automata as modeling formalisms has wide applicability—compact, expressive, and declarative representations of discrete dynamic systems are particularly suitable for environments in which agents must reason and act in previously unseen domains with previously unseen constraints. In these settings, descriptions of new worlds and goals must be rapidly formulated and efficiently transmitted to agents before they act. The General Game Playing (GGP) domain provides one such setting in which Cell Automata may be used to model the physics of multiplayer games [5]. The GGP framework, based on the discrete automata defined in this paper, has been used in four competitions at AAAI (see, for example, [6]), and has provided a basis for research in transfer learning [7] and game playing [8] [9]. Semantic Enterprise Management—the study of discrete dynamic systems described by declarative business rules using propositional logic–provides another environment where these automata may prove particularly useful.

2 Propositional Automata

A Propositional Automaton is a mathematical representation of a circuit-based formalism for discrete dynamic systems. The dynamics of a Propositional Automaton are described by a Propositional Net, an example of which is shown in Figure 1. This net defines the synchronous computation of the function:

$$L_{1,1} = (L_{1,1} \wedge (\neg Forward \vee Bump)) \vee (L_{1,2} \wedge (FacingDown \wedge Forward)) \vee (L_{2,1} \wedge (FacingLeft \wedge Forward))$$

A **Propositional Net** is a network of boolean gates and latches, along with transitions that control the flow of information between those components.

Definition 1 (Propositional Net). *A **propositional net** is a 4-tuple,* $\langle P, G, T, w \rangle$. ***P** is a finite set of* propositions. *A* proposition, p, *is a boolean-valued latch, with domain,* $dom(p) = \{\mathbf{0}, \mathbf{1}\}$. ***G** is a finite set of* boolean gates. *A* boolean gate, g, *is a function,* $g : \{\mathbf{0}, \mathbf{1}\} \times \{\mathbf{0}, \mathbf{1}\} \mapsto \{\mathbf{0}, \mathbf{1}\}$. ***T** is a finite set of* transitions. *A* transition, t, *is an*

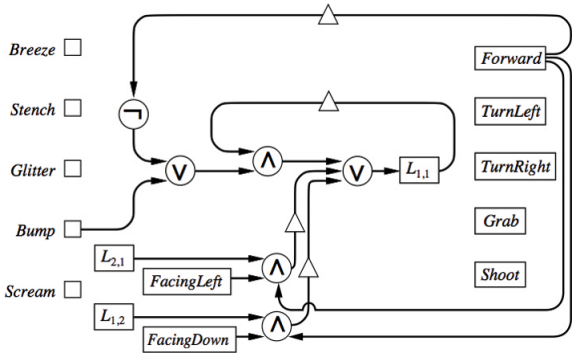


Fig. 1. A Propositional Net that represents a subset of the physics of the Wumpus World [1]

identity function, $t : \{\mathbf{0}, \mathbf{1}\} \mapsto \{\mathbf{0}, \mathbf{1}\}$. ***w*** *is a* connectivity function*:* $w : \{P \mapsto T \cup G \cup \{\emptyset\}; G \mapsto P \times P; T \mapsto P\}$. *Along with* P*,* G*, and* T*,* w *defines a graph in which cycles (if they exist) contain at least one transition. Where* w *returns a pair, the notation* $w(\cdot)_i$ *is used to index into that pair.*

For the sake of clarity, two simplifications have been made to the Propositional Net in Fig. 1: The definition of w implies that a Propositional Net is a bipartite graph with two sets of vertices, P and $(G \cup T)$, while boolean gates in the figure are connected in series. The definition of G requires that a boolean gate take exactly two inputs, while gates in the figure take one and three. These superficial discrepancies can be reconciled by adding of a constant number of propositions and gates.

Definition 2 (Proposition Partition). *The propositions in a Propositional Net can be partitioned into three disjoint subsets based on the definition of* w*.* P_I *is the set of* input propositions*,* p*, for which* $w(p) = \emptyset$*.* P_B *is the set of* base propositions*,* p*, for which* $w(p) \in T$*.* P_V *is the set of* view propositions*,* p*, for which* $w(p) \in G$*.*

Definition 3 (Input, Base, and View Marking). ***Markings*** *are functions that map propositions to boolean values. An* ***input marking*** *is a total function,* $m_I : P_I \mapsto \{\mathbf{0}, \mathbf{1}\}$*.* ***Base and view markings****,* m_B *and* m_V*, are defined similarly.*

Definition 4 (View Marking Computation). *Given an input marking,* m_I*, and a base marking,* m_B*, a view marking,* m_V *is computed as follows: Let* m_V *be an empty function* $m_V : P_V \mapsto \{\mathbf{0}, \mathbf{1}\}$*. Topologically sort the set of view propositions,* P_V*, such that for each* $p \in P_V$*, with* $g = w(p)$*,* $w(g)_1$ *and* $w(g)_2$ *(if they are view propositions) appear before* p*, ensuring that if the value of one view proposition depends on another, the latter is computed first. For all* $p \in P_V$*, in this sorted order, let* $g = w(p)$ *and* $m_V(p) = g(m(w(g)_1), m(w(g)_2))$*.*

Lemma 1 (View Uniqueness). *An input marking,* m_I*, and a base marking,* m_B*, define a unique view marking,* m_V*.*

Proof. Assume m_1 and m_2 agree on m_I and m_B, not on m_V: for some $p \in P_V$ $m_1(p) \neq m_2(p)$. Let $g = w(p)$. Then m_1 and m_2 must disagree on either $w(g)_1$

or $w(g)_2$; by assumption this disagreement must be on a view proposition. Because P_V is finite and w defines a graph in which any cycles include a transition, the assumption of disagreement on any view proposition requires disagreement on an input or base proposition, a contradiction.

Definition 5 (Marking). *A **marking** is a function, m, defined as the union of the three functions, m_I, m_B, and m_V, that represent the state of a Propositional Net.*

The **dynamics** of a Propositional Net—transitions between states—are defined in terms of the sets G and T, and the function w.

Definition 6 (Base Marking Update). *Given a marking, m, a new base marking, m'_B is computed as follows: for each base proposition, $p \in P_B$, let $t = w(p)$ and insert a mapping into m'_B, $p \mapsto t(m(w(t)))$.*

Lemma 2 (Update Closure). *Every marking m defines a unique valid base marking update.*

Proof. A base marking, m_B, is a total function from P_B to $\{\mathbf{0}, \mathbf{1}\}$. A marking, m, is a total function from P to $\{\mathbf{0}, \mathbf{1}\}$. Base marking update is defined as the definition of a function in which each base proposition, $p \in P_B$, is mapped to $m(p')$, where $p' \in P$. Thus, base marking update must produce a total function from P_B to $\{\mathbf{0}, \mathbf{1}\}$.

Theorem 1 (Update Complexity). *Given a base marking, m_B, the complexity of base marking update is $O(n^2)$, where $n = |P|$.*

Proof. Base marking update requires four steps: (1) Creating an input marking m_I by assigning a value to each input proposition requires $O(|P_I|) = O(n)$ time. (2) Sorting the view propositions (following Definition 4) requires $O(|P_V|^2) = O(n^2)$ time. (3) Performing the view marking computation requires $O(|P_V|) = O(n)$ time. (4) Performing the computation of Definition 6 requires $O(|P_B|) = O(n)$ time. Executing these steps in series requires $O(n^2)$ time. If the results of the sort are cached, complexity is reduced to amortized linear time.

A **Propositional Automaton** defines a physics in the form of a Propositional Net, an initial state, and a function for computing legal inputs to that net that provides constraints on agent inputs beyond the limitations dictated by the net itself.

Definition 7 (Propositional Automaton). *A **Propositional Automaton** is a triple $\langle N, l, M_B^0 \rangle$, where N is a Propositional Net; l is a* legality function*—a mapping from base markings to finite sets of input markings: $l : m_B \mapsto 2^{m_I}$; and M_B^0 is an initial base marking.*

3 Cell Automata

A Cell Automaton is a mathematical formalism with properties that are closely related to Propositional Automata. As Propositional Nets describe the dynamics of Propositional Automata, Cell Nets describe the dynamics of Cell Automata. An example of a Cell Net is shown in Figure 2. This net defines synchronous login verification: the

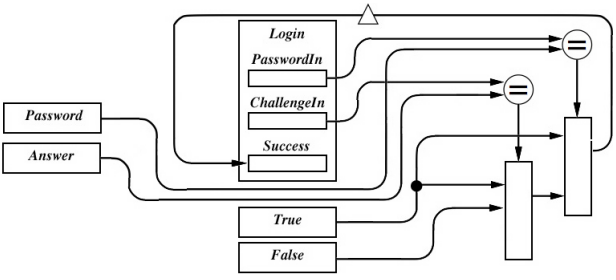


Fig. 2. A Cell Net that implements simple login verification. Simplifications made for clarity.

value of the latch, $Success$, is set to $True$ if either $PasswordIn = Password$ or $ChallengeIn = Answer$, and $False$, otherwise. The representation of this system as a Propositional Net would be significantly more convoluted. A **Cell Net** is a network of multi-valued latches, comparators and selectors along with transitions that control the flow of information between those components.

Definition 8 (Cell Net). *A **cell net** is a 6-tuple, $\langle O, C, E, S, T, w\rangle$. **O** is a finite set of* object constants*. **C** is a finite set of* cells*. A* cell*, c, is a multi-valued latch, with domain, $dom(c) = O$. **E** is a finite set of* comparators*. A* comparator *is a function, $e : O \times O \mapsto \{\mathbf{0}, \mathbf{1}\}$. A* comparator *returns* **1** *if its arguments are identical,* **0** *otherwise. **S** is a finite set of* selectors*. A* selector *is a function, $s : O \times O \times \{\mathbf{0}, \mathbf{1}\} \mapsto O$. A* selector *returns its first argument if its third is* **1***, its second otherwise. **T** is a finite set of* transitions*. A* transition*, t, is an identity function, $t : O \mapsto O$. **w** is a* connectivity function*: $w : \{C \mapsto T \cup S \cup \{\emptyset\}; E \mapsto C \times C; S \mapsto C \times C \times E; T \mapsto C\}$. Along with O, C, E, S, and T, w defines a graph in which any cycles contain at least one transition. Where w returns either a pair or a triple, the notation $w(\cdot)_i$ is used to index into that pair.*

For the sake of clarity, a simplification has been made to the Cell Net in Fig. 2. The definition of w implies that the inputs to a selector must be cells; the figure shows selectors connected directly to objects. This discrepancy is superficial and may be reconciled by the addition of a constant number of selectors, comparators, and transitions.

Definition 9 (Cell Partition). *The cells in a Cell Net can be partitioned into three disjoint subsets based on the definition of w. C_I is the set of* input cells*, c, for which $w(c) = \emptyset$. C_B is the set of* base cells*, p, for which $w(c) \in T$. C_V is the set of* view cells*, p, for which $w(c) \in S$.*

Definition 10 (Input, Base, and View Assignments). ***Assignments** are functions that map cells to object constants. An **input assignment** is a total function, $a_I : C_I \mapsto O$. **Base and view assignments**, a_B and a_V, are defined similarly.*

Definition 11 (View Assignment Computation). *Given an input assignment, a_I, and a base assignment, a_B, a view assignment, a_V is computed as follows:*

Let a_V initially be the incomplete function $a_V : C_V \mapsto O$, containing no mappings. Topologically sort the set of view cells, C_V, such that for all $c \in C_V$, with $w(c) = s$

and $w(s)_3 = e$, $w(s)_1$, $w(s)_2$, $w(e)_1$ and $w(e)_2$ (if they are view cells) appear before c. This ensures that if the value of one view cell depends on the value of another, the value of the latter is computed first. For each c, in order, let $s = w(c)$ and insert a mapping into a_V, $c \mapsto s(a(w(s)_1), a(w(s)_2), w(s)_3)$.

Lemma 3 (View Uniqueness). *An input assignment, a_I, and a base assignment, a_B, define a unique view assignment, a_V.*

Proof. The proof follows that of Lemma 1; disagreements between two assignments that agree on base and input assignments must be confined to the view assignments, but the finiteness of C_V means such disagreement results in a contradiction.

Definition 12 (Assignment). *An **assignment** is a function, a, defined as the union of the three functions, a_I, a_B, and a_V. The state of a Cell Net is represented by an assignment.*

The **dynamics** of a Cell Net—transitions between states—are defined in terms of the sets C, S, and T, and the function w.

Definition 13 (Base Assignment Update). *Given an assignment, a, a new base assignment, a'_B, is computed as follows: For each base cell, $c \in C_B$, let $t = w(c)$ and insert a mapping into a'_B, $c \mapsto t(a(w(t)))$.*

Lemma 4 (Update Closure). *Every assignment, a, defines a unique valid base assignment update.*

Proof. The proof follows that of Lemma 2.

Theorem 2 (Update Complexity). *Given a base assignment, a_B, the complexity of base assignment update is $O(n^2)$, where $n = |C|$.*

Proof. Base assignment update requires four steps: (1)An input assignment, a_I, must be created by assigning a value to each input cell. Doing so requires $O(|C_I|) = O(n)$ time. (2) The sort described in Definition 11 must be performed on the set of view cells. Doing so requires $O(|C_V|^2) = O(n^2)$ time. (3) The computation described in Definition 11 must be performed on the set of view cells. Doing so requires $O(|C_V|) = O(n)$ time. (4) The computation described in Definition 13 must be performed on the set of base cells. Doing so requires $O(|C_B|) = O(n)$ time. The execution of these steps in series requires $O(n^2)$ time; if the sort is cached, this can be reduced to amortized linear time.

A **Cell Automaton** defines a physics in the form of a Cell Net, an initial state, and a function for computing legal inputs to that net.

Definition 14 (Cell Automaton). *A **Cell Automaton** is a triple $\langle N, l, a_B^0 \rangle$. **N** is a Cell Net. **l** is a* legality function. *A* legality function *is a mapping from base assignments to finite sets of input assignments: $l : a_B \mapsto 2^{a_I}$. $\mathbf{a_B^0}$ is an initial base assignment to N.*

4 Comparison

Due to their graphical structure, Propositional and Cell Automata have properties closely related to digital logic circuits. Given the appropriate resources, simulation of Propositional and Cell Automata could be parallelized. However, the domains in which they

are used may not admit such an approach. While the design of intelligent agents might make use of parallelizable hardware, the simulation of dynamic, interactive systems is often implemented on single-processor machines. Accordingly, the proofs in this section assume serial, data-flow simulation, where the value of at most one structural element may be computed at a time, and the aim is to compare the most efficient encodings of Propositional and Cell Automata with respect to the most intelligent possible sequential simulation. In sections two and three we gave general complexity bounds for state update. Here we provide tighter bounds which hold only for nets that have been redesigned to allow for optimal data-flow simulation—only computing the values elements necessary for state update, and ignoring others. Although the process of optimizing a net may be difficult or computationally expensive, the computer architecture literature already contains techniques for doing so, and this optimization need only be performed once as a preprocessing step.

Cell Automata offer numerous advantages over Propositional Automata. Cell Automata are spatially less complex, containing exponentially fewer structural elements. Given the assumptions described above, Cell Automata are computationally less complex as well; in general, it is necessary to compute the value of fewer structural elements to perform state update. Finally, given any Propositional Automaton, there exists an algorithmic transformation to an equivalent, more efficient, Cell Automaton.

Lemma 5 (Expressiveness 1). *Cell Nets can represent arbitrary boolean functions.*

Proof. The following Cell Net describes the logical nand function. $N = \langle O, C, E, S, T, w\rangle$, where $O = \{\mathbf{0}, \mathbf{1}\}$; $C = \{x, y, not\_y, temp, nand\}$; $E = \{e_1, e_2\}$;$S = \{s_1\}$; $T = \{t_1\}$; and $w : \{x \mapsto \emptyset; y \mapsto \emptyset; temp \mapsto s_1; nand \mapsto t_1; e_1 \mapsto \langle y, \mathbf{0}\rangle; e_2 \mapsto \langle x, \mathbf{0}\rangle; s_1 \mapsto \langle \mathbf{1}, e_1, e_2\rangle; t_1 \mapsto temp\}$. Any boolean function can be represented by some combination of logical nand functions.

Lemma 6 (Equivalence 1). *Any Propositional Automaton can be represented by an equivalent Cell Automaton.*

Proof. Given a Propositional Automaton, $\langle N_p, l_p, m_B^0\rangle$, where $N_p = \langle P_p, G_p, T_p, W_p\rangle$, an equivalent Cell Automaton, $\langle N_c, l_c, a_B^0\rangle$, where $N_c = \langle O_c, C_c, E_c, S_c, T_c, w_c\rangle$, is defined as follows:

Define O_c to be the set $\{\mathbf{0}, \mathbf{1}\}$. For each proposition, $p \in P_p$, insert a cell, c_p, with domain, $dom(c_p) = \{\mathbf{0}, \mathbf{1}\}$, into C_c. For each transition, $t \in T_p$, insert a transition, t into T_c. For each gate, $g \in G_p$, insert an implementation of that gate as suggested by Lemma 4.1 into C_c, E_c, S_c and w_c. Denote that implementation G_g, with inputs G_g^{i1}, G_g^{i2} and output G_g^o. For each mapping $w_p : p \mapsto t$, insert a mapping into $w_c : c_p \mapsto t$. For each mapping $w_p : p \mapsto g$, insert a mapping into $w_c : c_p \mapsto G_g^o$. For each mapping $w_p : g \mapsto \langle p_1, p_2\rangle$, insert two mappings into w_c: $G_g^{i1} \mapsto c_{p_1}$ and $G_g^{i2} \mapsto c_{p_2}$. For each mapping $w_p : t \mapsto p$, insert a mapping into $w_c : c_p$. Convert l_p and m_B^0 into l_c and a_B^0 using the following transformation from markings to assignments: For each mapping $m : p \mapsto v$, insert a mapping into $a : c_p \mapsto v$.

Theorem 3 (Spatial Complexity 1). *Encoding a Propositional Automaton as an equivalent Cell Automaton may result in logarithmically fewer structural components.*

Proof. Given a Propositional Automaton, an equivalent Cell Automaton with $c = |C|$ cells and $o = |O|$ object constants contains a Cell Net that can represent up to o^c distinct states. While certain states might never be reachable given the dynamics of the Cell Net and the initial assignment to the Cell Automaton, the Propositional Automaton may have to contain at least $\log o^c = c \log o$ propositions.

Theorem 4 (Computational Complexity 1). *Given a Propositional Automaton, the most efficient encoding of an equivalent Cell Automaton for sequential simulation can implement comparison in logarithmically fewer computational steps.*

Proof. A Cell Automaton checks equality between cells c_1 and c_2, where $|dom(c_1)| = |dom(c_2)| = d$, in constant time given a comparator, e, and the mapping $w : e \mapsto \langle c_1, c_2 \rangle$. In an equivalent Propositional Automaton, the most spatially efficient encoding of c_1 and c_2 requires two sets of $\log d$ propositions; comparing their values requires $\log d$ comparisons.

Lemma 7 (Expressiveness 2). *Propositional Nets can represent comparators and selectors.*

Proof. Using combinations of boolean gates in the set G, Propositional Nets can represent arbitrary logical functions. A comparator for two sets of propositions, X and Y, implements the function $\wedge_i((X_i \wedge Y_i) \vee (\neg X_i \wedge \neg Y_i))$. A selector for two sets of propositions, X and Y, that uses a control bit, c, implements the function $\wedge_i((X_i \wedge \neg c) \vee (Y_i \wedge c))$.

Lemma 8 (Equivalence 2). *Any Cell Automaton has an equivalent representation as a Propositional Automaton.*

Proof. Given a Cell Automaton $\langle N_c, l_c, a_B^0 \rangle$, where $N_c = \langle O_c, C_c, E_c, S_c, T_c, w_c \rangle$, an equivalent Propositional Automaton, $\langle N_p, l_p, m_B^0 \rangle$, where $N_p = \langle P_p, G_p, T_p, W_p \rangle$, is defined as follows:

For each cell, $c \in C_c$, insert a set of $\log |O_c|$ propositions, P_c, into P_p. The set P_c will be used as a bitwise encoding of the $|O_c|$ objects in the Cell Automaton that may be held in the cell c . For example, given an assignment, a, where $a(c) = O_4$ (the fourth of the $|O_c|$ objects, in some fixed order), in the equivalent mapping, m, it will be the case that $m(P_{c1}) = \mathbf{1}$, $m(P_{c2}) = \mathbf{1}$, and $m(P_{ci}) = \mathbf{0}$ for all $i \in [2, |O_c|]$. For each comparator, $e \in E_c$, insert an implementation of a comparator as suggested by Lemma 6 designed to compare sets of size $\log |O_c|$ into P_p, G_p and w_p. Denote that implementation E_e, with $\log |O_c|$ inputs E_e^{i1}, $\log |O_c|$ inputs E_e^{i2} and output E_e^o. For each selector, $s \in S_c$, insert an implementation of a selector as suggested by Lemma 6 designed to select between sets of size $\log |O_c|$ into P_p, G_p and w_p. Denote that implementation S_s, with $\log |O_c|$ inputs S_s^{i1}, $\log |O_c|$ inputs S_s^{i2}, control input S_s^{ic} and $\log |O_c|$ outputs S_s^o. For each transition, $t \in T_c$, insert a set of $\log |O_c|$ transitions, T_c, into T_p. For each mapping $w_c : c \mapsto t$, for $1 \leq j \leq \log |O_c|$ insert a mapping into $w_p : P_{cj} \mapsto T_{cj}$. For each mapping $w_c : c \mapsto s$, for $1 \leq j \leq \log |O_c|$ insert a mapping into $w_p : P_{cj} \mapsto S_{sj}^o$. For each mapping $w_c : e \mapsto \langle c_1, c_2 \rangle$, for $1 \leq j \leq \log |O_c|$ insert two mappings into w_p: $E_{ej}^{i1} \mapsto P_{c_1 j}$ and $E_{ej}^{i2} \mapsto P_{c_2 j}$. For each mapping w_c :

$s \mapsto \langle c_1, c_2, e\rangle$, for $1 \leq j \leq \log|O_c|$ insert two mappings into w_p: $S^{i1}_{sj} \mapsto P_{c_1 j}$ and $S^{i2}_{sj} \mapsto P_{c_2 j}$. Additionally, insert the mapping $S^{ic}_s \mapsto E^o_e$. For each mapping $w_c : t \mapsto c$, for $1 \leq j \leq \log|O_c|$ insert a mapping into $w_p : T_{cj} \mapsto P_{cj}$. Convert l_c and a^0_B into l_p and m^0_B using the appropriate transformation from assignments to markings, respecting the bitwise encoding suggested in step 1: For each mapping $a : c \mapsto O_j$, for $1 \leq i \leq |O_c|$ if $i = j$ insert a mapping into $m : P_{ci} \mapsto \mathbf{1}$, otherwise insert the mapping $P_{ci} \mapsto \mathbf{0}$. For each mapping $a : c \mapsto O_1$, insert the mapping $m : P_{c1} \mapsto \mathbf{1}$ and the mappings $m_j : P_{cj} \mapsto \mathbf{0}$ for $1 \leq j \leq \log|O_c|$. Continue with the appropriate bitwise encodings for objects 2 through $|O_c|$.

Corollary 1. *Cell Automata are exactly as expressive as Propositional Automata.*

Theorem 5 (Spatial Complexity 2). *Given a Cell Automaton, there exists an equivalent Propositional Automaton with at most a logarithmic increase in the number of structural components.*

Proof. Suppose a Cell Automaton has $c = |C|$ cells, $o = |O|$ object constants, $e = |E|$ comparators, $s = |S|$ selectors, $t = |T|$ transitions, and the connectivity function w contains ω mappings. An equivalent Propositional Automaton, following the construction in Lemma 7, would contain p propositions, g boolean gates, τ transitions, and a connectivity function with σ mappings. Building comparators and selectors out of boolean gates for sets of propositions of size $\log o$ (following Lemma 7) results in the following numbers of components: **Propositions**: $p = c \log o + 4(e + s) \log o$. The first term comes from step 1 of Lemma 7. The second term comes from steps 2 and 3 of Lemma 7, where the construction of each of the e comparators and s selectors following Lemma 6 requires the introduction of $4 \log o$ intermediate propositions for comparing or selecting between sets of propositions of size $\log o$. **Boolean Gates**: $g = 4(e + s) \log o$, following the construction of the e comparators and s selectors in Lemma 6. **Transitions**: $\tau = t \log o$, from Lemma 7. **Connectivity**: $\sigma = \omega \log o + 4(e + s) \log o$, as each mapping in the connectivity function for the Cell Automaton requires $\log o$ mappings in the Propositional Automaton, and each gate introduced in Lemma 7 requires a mapping. Therefore, for a Cell Automaton with S_c structural components, including o objects, the equivalent Propositional Automaton following the construction of Lemma 7 has S_p structural components, where $S_p \leq 8 \cdot S_c \log o$.

Lemma 9 (Computational Complexity 2). *The computation of the value of any cell in a Cell Net requires, at worst, time linear in the sum of the sizes of the domains of the cells that that computation depends on.*

Proof. The value of a cell c will depend on the values of some set of cells, $\{c_1, \ldots, c_n\}$. That relationship can be encoded as a set of cascaded selectors, view cells, and comparators such that the longest path through the resulting structure traverses $\sum_{i=1}^{n} |dom(c_i)|$ selectors and comparators:

1. Let $i = 1$ and let $j = 1$.
2. Assign c_i the j^{th} value in $dom(c_i)$. Create a cell, c_{ij}, such that under any assignment, a, $a(c_{ij})$ is always equal to the j^{th} value in $dom(c_i)$. Create a comparator, e, along with the mapping, $w : e \mapsto \langle c_i, c_{ij}\rangle$. Create a selector, s, along with the mapping, $w : s \mapsto \langle c_1, c_2, e\rangle$, where c_1 and c_2 are created in steps (3) and (4).

3. If $i = n$
   (a) then create a cell, c_{ret}, such that under any assignment, a, $a(c_{ret})$ is always equal to the value that should correspond to the current assignment of $\{c_1, c_2, \ldots, c_n\}$.
   (b) otherwise create a view cell, c_v, along with a mapping, $w : c_v \mapsto s$, where s is the result of letting $i = i + 1$, $j = 1$, and going to step (2).
4. If $j = |dom(c_i)| - 1$
   (a) then create a cell identical to c_1.
   (b) otherwise create a view cell, c_v, along with a mapping, $w : c_v \mapsto s$, where s is the result of letting $j = j + 1$ and going to step (2).

Theorem 6 (Computational Complexity 3). *Given the most efficient encoding of a Cell Automaton for sequential simulation, the most efficient encoding of an equivalent Propositional Automaton performs state update in exponentially more computational steps.*

Proof. In a Cell Net, for any cell, c, whose value depends on the values of some set of cells, $\{c_1, \ldots, c_n\}$, the amount of time required to compute the value of c is $O(\sum_i |dom(c_i)|)$. In an equivalent Propositional Net, the most efficient encoding of those cells would require $p = \sum_i \log |dom(c_i)|$ propositions. Together, those propositions correspond to a logical function containing $O(2^p) = O(\prod_i |dom(c_i)|)$ disjuncts, each of which must be evaluated in series before producing a result. This increased computational cost must be applied across all sets of propositions corresponding to cells that require update.

Theorem 7 (Transformation). *Given a Propositional Automaton, there exists a transformation to an equivalent Cell Automaton that can perform state update in no more steps, and possibly in exponentially fewer computational steps.*

Proof. Given a Propositional Automaton, the value of each proposition, p, depends on the values of some set of propositions, $\{p_1, p_2, \ldots, p_n\}$ and require $O(2^n)$ computational steps to compute. By transforming into a Cell Automaton, following Lemma 8, then performing the optimization of Lemma 9, one produces a Cell Automaton that contains a cell c for every proposition p whose value can be computed in $O(n)$. In the degenerate case that each proposition depends only on the value of a single other proposition, the equivalent Cell Automaton provides no savings, but does not require additional computation.

5 Conclusion

Propositional and Cell Automata are equally expressive, powerful mathematical models for formalizing the behavior of discrete dynamic systems. In general, Cell Automata are more efficient than Propositional Automata. Given equivalent automata (each optimized for efficient sequential simulation), a Cell Automaton may contain logarithmically fewer structural elements and require exponentially fewer operations for state update. Fortunately, given any Propositional Automaton, there exists an algorithmic transformation to an equivalent, possibly more efficient Cell Automaton. When

space and computation concerns dominate, a Cell Automaton representation may be preferred. In other cases, transformation to a Propositional Automaton may reveal advantageous structural properties of a dynamic system—factoring out independent substructures or finding bottlenecks—and this can be performed with only a logarithmic increase in the size of the representation. Future work includes further development of techniques and applications for these transformations, as well as formal exploration of the advantages over existing formalisms for discrete dynamic systems noted in the introduction.

References

1. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson, London (2003)
2. Nowack, A.: A guarded fragment for abstract state machines. In: Proceedings of the ESSLLI 2004 Workshop on Guarded Fragments (2004)
3. Andrews, T., et al.: Business process execution language for web services (2003)
4. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice Hall PTR, Upper Saddle River (1981)
5. Genesereth, M.R., Love, N., Pell, B.: General game playing: Overview of the AAAI competition. AI Magazine 26(2), 62–72 (2005)
6. Schiffel, S., Thielscher, M.: Fluxplayer: a successful general game player. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (2007)
7. Banerjee, B., Stone, P.: General game learning using knowledge transfer. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (2007)
8. Kaiser, D.M.: Automatic feature extraction for autonomous general game playing agents. In: Proc. of the 6th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems (2007)
9. Reisinger, J., Bahceci, E., Karpov, I., Miikkulainen, R.: Coevolving strategies for general game playing. In: Proc. of the 3rd IEEE Symp. on Computational Intel. and Games (2007)

Constructing Web Corpora through Topical Web Partitioning for Term Recognition

Wilson Wong, Wei Liu, and Mohammed Bennamoun

School of Computer Science and Software Engineering
University of Western Australia
Crawley WA 6009
{wilson,wei,bennamou}@csse.uwa.edu.au

Abstract. The need for on-demand discovery of very large, incremental text corpora for unrestricted range of domains for term recognition in ontology learning is becoming more and more pressing. In this paper, we introduce a new 3-phase web partitioning approach for automatically constructing web corpora to support term recognition. An evaluation of the web corpora constructed using our web partitioning approach demonstrated high precision in the context of term recognition, a result comparable to the use of manually-created local corpora.

1 Introduction

Major ontology learning tasks such as term recognition and relation discovery all rely heavily on the availability of large text corpora. Many current research either uses manually-constructed static *local corpora* with limited size, or subconsciously using the World Wide Web (the Web) as a corpus in an ad-hoc manner to serve the needs of ontology learning. For clarity, we define a local corpus as *"a semi-static, centralised and organised collection of clean text whose growth is characterised by infrequent, manual efforts."*. The rarity of large local corpora across various domains has tremendous influence on the applicability and adaptability of existing systems to a wider range of applications. In addition, the mediocre size and coverage of local corpora, and their inability to incorporate frequent changes in the domain all have impact on the performance of ontology learning.

Despite issues such as text quality and sampling, the Web remains the key to our problem of automatic and on-demand construction of very large corpora. Existing research on corpus construction using the Web is focused primarily on building domain corpora by downloading and cleaning web documents provided by search engines. While such collections have often been misconstrued as *web corpora*, they are in fact *web-based local corpora*. A web corpus is *"a live, decentralised collection of noisy text (e.g. web documents) which is subjected to continuous, autonomous and haphazard updates"*, and collections of documents downloaded to a central location definitely do not qualify as web corpora. Even though web-based local corpora are obtained from the Web, their contents are

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 67–78, 2008.

no longer distributed and dynamic. As research on ontology learning progresses, the need for on-demand construction of very large, dynamic web corpora for unrestricted range of domains becomes more and more pressing. Despite such urgency and the excitement associated with the Web offering indefinite amount of data across most genres of information, it is rare to find work that takes a step back to investigate how we can construct dynamic corpora from the Web.

In this paper, we propose a way of automatically constructing domain corpora on-demand to benefit the various phases of ontology learning, and more specifically, term recognition. We introduce an automatic *topical web partitioning* approach for generating web domain and contrastive corpora required for term recognition. The main contributions of this paper are 1) the development of a new practical approach for automatically constructing web corpora, and 2) the demonstration of high performance term recognition using web corpora. Section 2 summarises existing work on corpus construction, and related techniques in web technologies for constructing web corpora. In Section 3, we outline our new web partitioning approach and the associated techniques. In Section 4, we evaluate the web corpora constructed using our approach in the context of term recognition. We end this paper with an outlook to future work in Section 5.

2 Related Work

Current work in the literature focuses primarily on constructing web-based local corpora. One of such systems is *BootCat* [1]. This system constructs a web-based local corpus by downloading and cleaning the top n web pages returned by search engines using a set of seedterms. In addition to relying on seedterms, Agbago & Barriere [2] proposed a knowledge-richness estimator that takes into account semantic relations to support the construction of web-based local corpora. Documents containing both the seedterms and the desired relations are considered as better candidates to be included in the corpus. The candidate documents are ranked and manually filtered based on several term and relation richness measures. Baroni & Bernardini [3] offer a comprehensive compilation of work on web-based local corpora. To the best of the authors' knowledge, there is no literature on constructing actual web corpora without downloading web pages into local repositories. Therefore, we examine techniques from three related areas, namely, *web page clustering and classification*, *usage-based URI clustering*, and *web page importance ranking* to provide basis for our new approach.

In the first research area, web page clustering involves the discovery of naturally occurring groups of related web pages, while web page classification assigns web pages to predefined classes. Despite their minor functional differences, the conventional techniques for both clustering and classification of web pages rely on either the content, structure or both to enable the use of feature-based similarity to determine relatedness. For example, Estruch et al. [4] presented a *decision tree-based algorithm* for web page classification. Both web page content and the connection between web pages through hyperlinks are utilised to construct feature sets. The symmetric difference between feature sets is employed as the

distance metric for the partitioning of the decision tree. Crabtree et al. [5] proposed a variant of *suffix tree clustering* known as the *extended suffix tree clustering* for clustering web pages based on their common contents. More importantly, the authors introduced a new cluster selection algorithm for identifying the most useful clusters of web pages to be returned to the users. The selection algorithm is based on the heuristic that useful clusters should have minimal overlap and maximal coverage. The second related area is the usage-based clustering of *Uniform Resource Identifiers (URI)* for web personalisation and navigation. In this area, the navigation history of users are employed for clustering URIs. As such, the resulting clusters are URIs related through user navigational patterns instead of content similarity or site authority. Mobasher et al. [6] employed *association rule hypergraph partitioning* to cluster URIs based solely on their co-occurrence across different user sessions. As such, no computation of similarity or distance is required. The authors argued that conventional clustering techniques that require user sessions for computing feature-based similarity are not feasible since the size of navigation history is often very large. Lastly, the problem of determining the importance of web pages has been studied extensively in the area of link analysis. Several algorithms have been developed for the sole purpose of ranking web pages based on link structure. Most algorithms, such as the famous PageRank [7], assign numerical weights to each web page based several factors such as in-links to determine the page's relative importance on the Web.

3 Topical Web Partitioning

In general, we can view the construction of web domain corpora as a process of *web partitioning*. Given that U is the universal set of all web pages on the surface Web, web partitioning on set U creates two complementary sets $W = \{p_1, ..., p_n\}$ and $\overline{W} = \{p_1, ..., p_m\}$ where p_i is a web page and $n + m = |U|$. W is the set of all web pages from U whose content has been objectively determined based on certain criteria as suitable for characterising the domain of interest D. On the other hand, $\overline{W}$ is the set of all remaining web pages from U which are not in W. In theory, the web pages in $\overline{W}$ should have poor characterisation ability for D. Our domain characterisation criteria for web pages are 1) the popularity of a web page with respect to D, 2) the extent to which the vocabulary supported by a web page is inclined towards D, and 3) the extent to which the vocabulary of a web page is specific to D. From the perspective of corpus linguistics, these three criteria allow us to equate the set W to a web domain corpus since the textual content, both lexical and semantic, of the web pages in W is biased towards a single domain D.

The existing techniques in Section 2 are inadequate for constructing web domain corpora that satisfy our stringent domain characterisation criteria. The computational power required for both textual and link analysis in clustering all web pages on the Web makes such web partitioning approach infeasible. A more feasible way is through the selection of web pages based on their importance or relevance. However, the approach of simply querying web search engines using

seedterms and treating the search results as elements of W does not satisfy our second and third criteria. We cannot be certain that the top-ranked web pages obtained from search engines have high vocabulary coverage and contain specific vocabulary. On that basis, we propose a novel 3-phase web partitioning approach. The final output of the approach is a virtual, distributed and dynamic domain text collection (i.e. web domain corpus) which consists of a set of web site URIs, M whose content can be accessed as required over the Web. The detail of the three phases summarised above is presented in the following subsections.

3.1 Probabilistic URI Filtering

In this first phase, we use the seedterms w to construct a query which is later submitted to the Google search engine. The set of web page URIs G, returned by Google is processed to obtain only the base URIs, which is the part starting from the scheme up to the authority segment. Collectively, the distinct base URIs form a new set J. Each element $u_i \in J$ has three pieces of important information, namely, a popularity rank r_i, the number of web pages containing w, n_{wi}, and the total number of web pages $n_{\Omega i}$. We consider the PageRank attached to each web site by Google as a measure of popularity[1]. As for the second (i.e. n_{wi}) and third pieces of information (i.e. $n_{\Omega i}$), we need to perform additional queries with special search operators using Google. We obtain the number of web pages or pagecount by limiting the search to individual sites $u_i \in J$ (i.e. site search) using the operator *"site:"*. The pagecount of w can be achieved using the query *"w site:u_i"* while the total pagecount for site u_i can be estimated using the query *"g site:u_i"* where g is a string of function words separated by *"OR"*.

Next, we introduce three odds to satisfy our three domain characterisation criteria. These odds are the *Odds of Popularity (O_P)*, the *Odds of Vocabulary Coverage (O_C)*, and the *Odds of Vocabulary Specificity (O_S)*. We introduce a final measure known as the *Odds of Domain Characterisation (OD)*, which is the sum of all the three odds. The derivation of OD has been excluded due to space constraint. To obtain O_P, we need to define the distribution for computing the probability that web site u_x is popular with respect to w. We can employ the popularity rank r_i associated with each $u_i \in J$. It has been demonstrated that the various indicators of a web site's popularity such as the number of in-links, the number of out-links and the frequency of visits, follow the *Zipf's ranked distribution* [8]. As such, the probability that the site u_x with pagerank r_x is popular with respect to w can be defined using the probability mass function $P(r_x; |J|) = \frac{1}{r_x H_{|J|}}$ where $H_{|J|}$ is the $|J|$-th generalised harmonic number computed as $H_{|J|} = \sum_k^{|J|} \frac{1}{k}$. We can then compute O_P as $O_P = \frac{P(r_x;|J|)}{1-P(r_x;|J|)}$. In regard to the odds of vocabulary coverage O_C, we can estimate the potential of site u_x in covering the vocabulary of the domain in terms of the probability of encountering a web page from site u_x among all other web pages from all other sites in J that contain w. Intuitively, the more web pages from site u_x that

[1] Larger numerical value of r_i indicates higher rank.

contains w, the likelier it is that u_x has good coverage of the vocabulary of the domain represented by w. As such, this factor requires a cross-site analysis of pagecount. In this case, we consider the collection of all web pages from all sites in J that contain w as the sample space. The size of that space is n_w. Hence, we can compute the probability of encountering web pages from site u_x that contain w as $f_c(n_{wx}) = \frac{n_{wx}}{n_w}$ where n_{wx} is the number of web pages from the site u_x containing w. We can then compute O_C as $O_C = \frac{f_c(n_{wx})}{1-f_c(n_{wx})}$. The last odds O_S which attempts to capture the vocabulary specificity of web pages on a site, helps us to identify overly general sites. While a site may have high vocabulary coverage, this does not immediately qualifies it as one that appropriately characterises the domain represented by w. The vocabulary specificity of a site can be estimated using the variation in the pagecount of w from the total pagecount of that site. Within a single site with fixed total pagecount, an increase in the number of web pages containing w implies a decrease of pagecount not containing w. In such cases, a larger portion of the site would be dedicated to discussing w and the domain represented by w (i.e more specific vocabulary). As such, the examination of the specificity of vocabulary is confined within a single site, and hence, is defined over the collection of all web pages within that site. Formally, the function that provides the probability of encountering a web page containing w from site u_x out of all web pages of u_x is defined as $f_s(n_{wx}) = \frac{n_{wx}}{n_{\Omega x}}$. We can then compute O_S as $O_S = \frac{f_s(n_{wx})}{1-f_s(n_{wx})}$.

In order to filter out sites with poor domain characterisation ability, a threshold for OD, represented as OD_T, needs to be derived automatically as a combination of the individual thresholds related to O_P, O_C and O_S. Depending on the desired output, these individual thresholds can be determined using the combination of the averages and standard deviations related to O_P, O_C and O_S. The discussion on the derivation of OD_T is beyond the scope of this paper. It suffices to know that all sites $u_i \in J$ with their odds $OD(u_i)$ more than OD_T are considered as suitable candidates for characterising the domain represented by w. These suitable sites form a new set $K = \{u_i | (u_i \in J) \wedge (OD(u_i) > OD_T)\}$ for subsequent processing.

3.2 URI Clustering

This second phase serves two purposes, namely, to refine the web site URIs in K, and to discover internal aspectual polarisation of the web site URIs. The first phase establishes that the sites in K have high odds of popularity, and vocabulary coverage and specificity with respect to the domain represented by w. However, within K, the sites can characterise the domain in different aspects. We refer to this phenomenon as *domain aspect characterization*. For example, in the domain represented by *"molecular biology"*, the sites in K may address aspects such as *"products and services"*, *"techniques and tools"*, *"background references"*, *"novice reading materials"* and *"academic and research writings"*. Hence, it is necessary to identify and further filter out overly general web sites in K that are not suitable for constructing web domain corpus.

We employ a 2-pass term clustering algorithm known as the *Tree-Traversing Ants (TTA)* [9] together with featureless similarity measures to cluster the sites in K and later remove the overly general web sites. The similarity between any two elements (e.g. URIs, terms) to be clustered is given by [9]:

$$sim(x, y) = 1 - NGD(x, y)\theta \tag{1}$$

where θ is a constant for scaling the distance value by NGD [10], the featureless distance measure. The sheer size of the Web allows NGD to induce adequate statistical evidence for estimating relatedness without relying on any static resources. The first pass works in a manner similar to divisive clustering to create a tree structure where each leaf node $r_i \in LN$ in the tree corresponds to one cluster C_i. LN is the set of all leaf nodes created by the TTAs. The first pass, whose algorithm is available in Wong et al. [9] employs NGD for similarity measurement. A second pass, summarised in Algorithm 3 of Wong et al. [11], is then used to refine the output from the first pass by relocating leaf nodes with single term (i.e. isolated term). This second pass is functionally similar to agglomerative clustering, and employs two dedicated cluster relatedness measures based on Equation 1. The conditions which determine the consolidation of two leaf nodes are (1) high inter-cluster (i.e. inter-node) similarity between r_x and r_y, or $S(r_x, r_y)$, and (2) low intra-cluster (i.e. intra-node) variation of the combined r_x and r_y, or $V(r_x, r_y)$. The first condition ensures that the two nodes r_x and r_y have high similarity, while the second condition makes sure that the combination of r_x and r_y results in a *compact node*. The inter-cluster similarity (i.e. the group-average similarity) and the intra-cluster variation are defined in Equation 6 and Equation 7 of Wong et al. [11], respectively. During the second pass, the ants utilise the inter-cluster similarity $S(r_x, r_y)$ and the intra-cluster variation $V(r_x, r_y)$ as their measures for neighbourhood density. An ant carries each leaf node and senses for neighbourhood density with all other leaf nodes in LN. The merging is performed if the neighbourhood density exceeds a certain threshold $S2_T$. The neighbourhood density of leaf node r_x with every other leaf nodes $r_i \in LN$ is defined in Equation 8 of Wong et al. [11]. Intuitively, a high neighbourhood density is obtained if the pair r_x and r_i has high inter-cluster similarity and low intra-cluster variation. Finally, the decision on whether to combine a leaf node r_x with another leaf node r_y depends on Equation 10 of Wong et al. [11].

In this pass, set K is partitioned into clusters where each URI cluster characterises a certain aspect of the domain represented by seedterms w. In order to minimise human intervention during the process of URI clustering, the thresholds for the first pass, $S1_T$ [9] and the second pass, $S2_T$ [11] are determined automatically using the numerical descriptors related to the respective neighbourhood functions. Finally, after the first and second pass of clustering using the automatically determined thresholds $S1_T$ and $S2_T$, we obtain a new set $L = \{C_i | C_i = \{u_j | (u_j \in K) \wedge (u_j \text{ can only belong to one } C_i)\}\}$ from the original set of all URIs K. It is worth noting that the TTA algorithm performs hard clustering. In other words, each site u_i can only belong to one cluster.

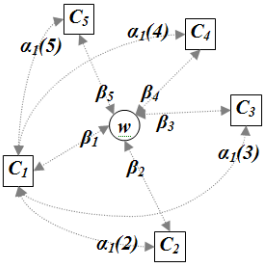


Fig. 1. An example layout of clusters, and their β- and α-scores

3.3 URI Cluster Selection

For this last phase, we employ two types of elementary relatedness score for identifying and removing URI clusters in L that have poor domain characterisation quality. The first type is the relatedness between the URI clusters and the seedterms w, or β-score. The second type is the inter-cluster relatedness between pairs of URI clusters, or α-score. It is worth pointing out that for this phase to work, the size of L has to be larger than 2. The degree of relatedness between a cluster of URIs $C_x \in L$ and the seedterms w is defined as $\beta_x = \frac{\sum_{u_i \in C_x} sim(w,u_i)}{|C_x|}$ where $sim(w, u_i)$ is as defined in Equation 1. For brevity, we refer to this intra-cluster with seedterm relatedness for cluster C_x as the β-score of C_x. Next, the α-score is based on the heuristic that:

Heuristic 1. *Among all web pages on the surface Web, the web pages hosted by the majority of the web sites in set L are the most suitable to characterise the domain represented by w.*

From Heuristic 1, we know that clusters containing the least suitable URIs will exhibit inconsistencies in terms of their relatedness with other clusters containing more suitable URIs. To illustrate the notion of relatedness inconsistency, assume that we have five clusters as shown in Figure 1. Intuitively, if the α-score of C_1 is high with respect to all other clusters $\{C_2, C_4\}$ except cluster $\{C_3\}$, then the possibility of C_3 being a poor characteriser of the domain becomes higher. In other words, C_1 is better at characterising the domain represented by w than C_3. To capture such relatedness inconsistency, we define an inter-cluster relatedness score between (C_x, C_i) using the group-average similarity defined in Equation 6 of Wong et al. [11] as $\alpha_x(i) = S(C_x, C_i)$. We refer to this inter-cluster relatedness between C_x and C_i as the α-score of C_x given C_i.

Next, we derive two new scores based on the two elementary scores (i.e. α- and β-score). We assign a score for each $C_i \in L$ based on the property of relatedness inconsistency between clusters. This first derived score, defined as the *neighbourhood score* or N-score, captures the extent to which a cluster C_x has strong inter-cluster relatedness with its neighbouring clusters. As we have pointed out from Heuristic 1 and the associated justifications, the more clusters

with which C_x has strong relatedness, the better chances it is that C_x contains URIs that are suitable for characterising the domain. We define the notion of *"strong relatedness"* as clusters having above average α-score with C_x. Such clusters form a temporary set R_x associated with cluster C_x. The set R_x is defined as $\{C_i|(C_x \neq C_i) \wedge (\alpha_x(i) > E[\alpha_x])\}$ where $E[\alpha_x]$ is the average α-score of C_x with all its neighbouring clusters. Next, each cluster C_x is assigned an N-score, which is a normalised α-score corrected by the β-score of the associated clusters in R_x. We define the N-score for cluster C_x as:

$$\mathbf{N}_x = \sum_{C_i \in R_x} \beta_i(\alpha_x(i) - E[\alpha_x]) \tag{2}$$

With N-score, the status of C_x being a suitable domain characteriser is further strengthened if the clusters which are already highly related to C_x also have strong associations with the seedterm. In addition to the N-score, a second derived *individual score* or I-score is necessary to allow the adjustment of the amount of contribution of α-score and β-score during cluster selection. I-score combines β-score and the average inter-cluster relatedness of a cluster, $E[\alpha_x]$. The I-score of cluster C_x is defined as:

$$\mathbf{I}_x = (\beta_x)^{\epsilon(1-\lambda_L)}(E[\alpha_x])^{\epsilon\lambda_L} \tag{3}$$

where ϵ is a constant scaling factor for λ_L. λ_L is the significance adjustment factor that controls the extent of contribution of β-score and the average α-score towards the computation of I-score. It is important to note that the values of β_x and $E[\alpha_x]$ fall within $[0, 1]$. λ_L is defined as $\lambda_L = \frac{SD[\beta]}{|L|}$. λ_L is a function of $|L|$ (i.e. the number of clusters produced in the previous phase using TTA), and the standard deviation of the β-score, $SD[\beta]$.

Finally, we combine the N-score and I-score of cluster C_x to obtain its Θ-score as $\Theta_x = \mathbf{N}_x\mathbf{I}_x$. The Θ-score is a heuristically-derived cluster selection score that optimises the discriminative property of the two types of relatedness considered for identifying URI clusters with high domain characterisation ability. Higher Θ-score implies greater domain characterisation ability. Instead of manually providing thresholds, which vary between different L, for determining the acceptance or rejection of clusters in L, we employ the average and standard deviation of the Θ-score to define a cut-off point Θ_T. All clusters that demonstrate Θ-scores higher than the threshold Θ_T are accepted as members of a new set $M = \{C_i|(C_i \in L) \wedge (\Theta_i > \Theta_T)\}$.

4 Evaluations and Discussions

The aim of this evaluation is to demonstrate the applicability of the web domain corpora constructed on-demand using our web partitioning approach in the context of *term recognition* by contrasting the assessment against the use of local domain corpora. We prepared two local collections of text, namely, a local contrastive corpus of about $6,700,000$ word count consisting of online news articles

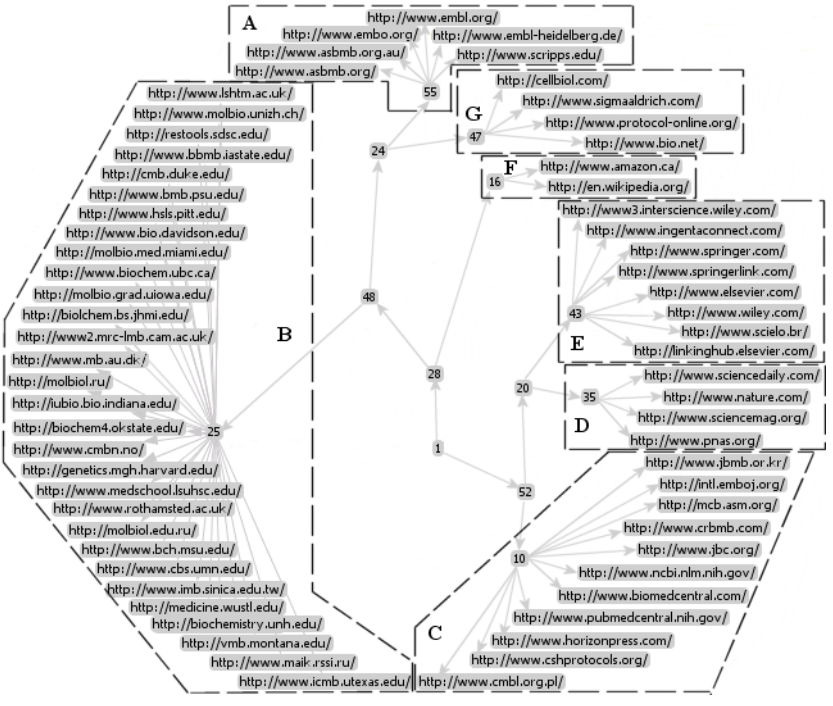


Fig. 2. The output of automatic URI clustering. There are seven clusters, each representing a certain aspect of the *"molecular biology"* domain. Cluster A represents the different societies or organisations for life sciences. Cluster B contains the URIs of biomedical schools, while cluster C represents various life sciences publishers and archives. Cluster D consists of URIs of scientific periodicals and magazines. Cluster E contains the URIs of academic publishers. Clusters F and G are groups of URIs of other resources in the general domain and the life sciences domain, respectively.

which span across many genres, and a local domain corpus for the domain of *"molecular biology"* known as the *GENIA* corpus [12] which consists of about 400,000 words. In addition, we constructed a web domain corpus and a web contrastive corpus using our web partitioning approach described in Section 3 based on the seedterm w=*"molecular biology"*. We will discuss the construction process in the next paragraph. The term candidates used in this evaluation were extracted from the GENIA corpus. The GENIA corpus is annotated with both part-of-speech and semantic categories. A gold standard for the *"molecular biology"* domain can be constructed by extracting the terms which have semantic descriptors enclosed by `cons` tags. For practicality reasons, we have limited the set of term candidates to 2,927. We performed term recognition using a measure known as the *Odds of Termhood (OT)* [13]. *OT* is a probabilistically-derived measure for scoring and ranking term candidates for term recognition. *OT* is a combination of seven evidence based on qualitative term characteristics, founded on formal models of word distribution. A detail discussion on the *OT* measure

B_j^X	X=OT_WC					X=OT_LC				
	pre	rec	$F_{0.1}$	F_1	acc	pre	rec	$F_{0.1}$	F_1	acc
j=1	92.50	16.50	65.19	28.00	34.98	97.00	17.30	68.36	29.36	36.21
j=2	88.13	31.43	75.71	46.34	44.21	93.63	33.39	80.44	49.23	47.22
j=3	87.67	46.90	81.25	61.11	54.25	90.42	48.37	83.80	63.03	56.51
j=4	86.31	61.57	83.27	71.87	63.07	88.25	62.95	85.14	73.48	65.19
j=5	83.95	74.86	83.03	79.14	69.76	86.70	77.31	85.75	81.73	73.52
j=6	81.38	87.07	81.86	84.13	74.82	84.29	90.19	84.80	87.14	79.60
j=7	76.75	95.81	78.16	85.23	74.55	79.43	99.15	80.89	88.20	79.67
j=8	76.63	100.00	78.29	86.77	76.63	76.63	100.00	78.29	86.77	76.63

Fig. 3. The performance of term recognition with 2,927 term candidates based on the measure OT using the local corpora, and the web corpora constructed with our web partitioning approach. The results using local corpora and web corpora are shown in the columns OT_LC and OT_WC, respectively. The result is obtained through the cumulative binning of the ranked terms in sizes of $j \times 400$ using the GENIA gold standard. The values of the performance measures in darker shades are the best performing ones. The columns *pre*, *rec*, $F_{0.1}$, F_1 and *acc* correspond to precision, recall, $F_{0.1}$-score, F_1-score and accuracy, respectively.

is beyond the scope of this paper. It suffices to know that the measure takes an input, which is a term candidate a, and produces a score as the output for ranking.

To construct our web corpora, we utilised the search term w=*"molecular biology"* to obtain the top 700 URIs returned by Google search engine. A set of 665 unique base URIs, J was produced. Using the mean of O_C and O_S, and the minimum of O_P to compute the threshold OD_T for the probabilistic URI filter described in Subsection 3.1, we obtained a set of 65 most suitable web site URIs, denoted as K for representing the *"molecular biology"* domain. The choice of the individual thresholds is motivated by placing more emphasis on vocabulary coverage and specificity than popularity. The size of K can be increased by lowering the thresholds of O_C and O_S. In the second phase described in Subsection 3.2, we set the first-pass clustering threshold $S1_T$ as the average of the pair-wise similarity sim defined in Equation 1. To compute the threshold $S2_T$, we utilised the average plus standard deviation of the inter-cluster similarity $S(r_x, r_y)$, and the mean of the intra-cluster variation $V(r_x, r_y)$. Seven clusters were created from the 65 URIs, and are collectively referred to as set L as shown in Figure 2. Generally, there is a strong tendency for each of these clusters to represent a certain aspect of the *"molecular biology"* domain. This can be determined by examining the theme of the web sites. The seven clusters undergo a cluster selection process described in Subsection 3.3 as the last phase of web partitioning. Using the average Θ-score as the threshold, clusters E and F were rejected. The final output set M consists of only five clusters with 55 URIs. These 55 web sites have over 22 million web pages in total and are now part of the web domain corpus W for *"molecular biology"*. The remaining web pages on the surface Web contribute to the web contrastive corpus $\overline{W}$.

Figure 3 shows the performance of term recognition using the 2,927 term candidates. The result using web corpora and local corpora is shown in the column OT_WC and OT_LC, respectively. Each row is a bin, of size $j \times 400$, containing the ranked term candidates which were sorted according to the scores assigned by OT (i.e. higher score, higher rank). In other words, the first bin $B_1^{OT\_LC}$ contains the top 400 term candidates ranked according to OT_LC, while the fourth bin $B_4^{OT\_WC}$ holds the top 1,600 term candidates ranked by OT_WC. The smaller cells under the columns OT_WC and OT_LC are performance indicators calculated using the gold-standard that comes with the GENIA corpus. Using the first bin $B_{j=1}^{X}$ (i.e. first 400 ranked terms) as an example, we can observe from Figure 3 that term recognition using local corpora with the measure OT (i.e. $X = OT\_LC$) achieved a precision of 97%, while the use of web corpora (i.e. $X = OT\_WC$) scored a 92.5% precision. The values of the performance measures in darker shades are the best performing ones. For instance, the best $F_{0.1}$ score achieved through the use of web corpora is 83.27% at bin $j = 4$, which is just 2% less than the $F_{0.1}$ at $B_4^{OT\_LC}$. In short, Figure 3 provides empirical evidence which demonstrates the ability of our web partitioning approach in constructing web domain corpora which are applicable to real-world applications. The figure shows that the results of term recognition using local corpora and web corpora are comparable, with slightly lower performances between $1 - 4\%$ across all indicators. Such performance is in fact promising considering the minimal effort required to produce the web corpora. Moreover, unlike texts in local corpora which have been vetted by human experts, the web corpora do not undergo any form of verification of text quality. This indicates the possibility that lack of text quality in a web corpus can indeed be compensated by its size, at least in the context of term recognition.

5 Conclusions

The applicability of many term recognition techniques remains limited due to the rarity of very large text corpora in the target domain and across contrasting domains. To address this problem, we introduced a novel 3-phase web partitioning approach for constructing web corpora on demand for term recognition. The advantages of web corpora are 1) the *evolutionary nature* of web corpora which automatically reflect changes in the domain, 2) the *unbounded volume* of web corpora, 3) the *comparable performance* of web corpora in the context of term recognition, and 4) the *significant savings* in effort during corpus construction. Overall, we have shown that 1) it is computationally feasible to construct web corpora for practical applications using our web partitioning approach, and 2) that our web corpora can be used to achieve favourable term recognition performance in comparison with high-quality, manually-crafted local corpora. Several future works have been planned which include the study on the effect of web partitioning using different seedterms w, and the research on examining how the content of web corpora evolve over time and its effect on term recognition. We are also planning to study the possibility of extending the use of web corpora

to other applications which requires contrastive analysis, and other tasks in ontology learning such as relation discovery.

Acknowledgement

This research was supported by the Australian Endeavour International Postgraduate Research Scholarship, the 2008 UWA Research Grant, and the Curtin Chemical Engineering Inter-University Collaboration Fund.

References

1. Baroni, M., Bernardini, S.: Bootcat: Bootstrapping corpora and terms from the web. In: Proceedings of the 4th Language Resources and Evaluation Conference (LREC), Lisbon, Portugal (2004)
2. Agbago, A., Barriere, C.: Corpus construction for terminology. In: Proceedings of the Corpus Linguistics Conference, Birmingham, UK (2005)
3. Baroni, M., Bernardini, S.: Wacky! working papers on the web as corpus. In: GEDIT, Bologna, Italy (2006)
4. Estruch, V., Ferri, C., Hernandez-Orallo, J., Ramirez-Quintana, M.: Web categorisation using distance-based decision trees. In: Proceedings of the International Workshop on Automated Specification and Verification of Web Sites, WWV (2006)
5. Crabtree, D., Gao, X., Andreae, P.: Improving web clustering by cluster selection. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, WI (2005)
6. Mobasher, B., Cooley, R., Srivastava, J.: Creating adaptive web sites through usage-based clustering of urls. In: Proceedings of the Workshop on Knowledge and Data Engineering Exchange (1999)
7. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report; Stanford University (1998)
8. Adamic, L., Huberman, B.: Zipfs law and the internet. Glottometrics 3(1), 143–150 (2002)
9. Wong, W., Liu, W., Bennamoun, M.: Tree-traversing ant algorithm for term clustering based on featureless similarities. Data Mining and Knowledge Discovery 15(3), 349–381 (2007)
10. Cilibrasi, R., Vitanyi, P.: The google similarity distance. IEEE Transactions on Knowledge and Data Engineering 19(3), 370–383 (2007)
11. Wong, W., Liu, W., Bennamoun, M.: Featureless data clustering. In: Song, M., Wu, Y. (eds.) Handbook of Research on Text and Web Mining Technologies. IGI Global (2008)
12. Kim, J., Ohta, T., Teteisi, Y., Tsujii, J.: Genia corpus - a semantically annotated corpus for bio-textmining. Bioinformatics 19(1), 180–182 (2003)
13. Wong, W., Liu, W., Bennamoun, M.: Determining termhood for learning domain ontologies in a probabilistic framework. In: Proceedings of the 6th Australasian Conference on Data Mining (AusDM), Gold Coast (2007)

An Ontology Formalization of Relation Type Hierarchy in Conceptual Structure Theory

Philip H.P. Nguyen[1], Ken Kaneiwa[2], Dan R. Corbett[3], and Minh-Quang Nguyen[4]

[1] Justice Technology Services, Dep. of Justice, Gov. of South Australia, Adelaide, Australia
nguyen.philip@saugov.sa.gov.au
[2] National Institute of Information and Communications Technology, Kyoto, Japan
kaneiwa@nict.go.jp
[3] Schafer Corporation, Arlington, Va., USA
daniel.corbett.ctr@darpa.mil
[4] Institut National de Recherche Scientifique, EMT Lab, Montreal, Canada
nguyenmq@emt.inrs.ca

Abstract. This paper presents an enhancement to ontology formalization, combining previous work in Conceptual Structure Theory and Order-Sorted Logic. In particular, the relation type hierarchy of the former theory is a subset of the predicate hierarchy of the latter. Most existing ontology formalisms place greater importance on concept types, but this paper focuses more on relation types, which are in essence predicates on concept types. New notions are introduced and new properties identified with the aim of completing missing arguments in relation types. The end result is a new ontology, that we call the closure of the original ontology, on which automated inference could be more easily produced (e.g., a query-answering system for legal knowledge).

1 Introduction

In a formalism based on Conceptual Structure Theory [2][8][9], an ontology is in essence a mapping of a real world into an abstract conceptual world, consisting of a concept type hierarchy, a relation type hierarchy, and formal relationships between them. This is similar to what is proposed by the popular Web Ontology Language (OWL) in which an ontology is defined as a collection of a set of classes (unary predicates), a set of properties (binary predicates), and a set of declarations describing how classes and properties are related [10]. Ontologies are usually distinguished from databases. An ontology represents shared and commonly-agreed-to knowledge while a database contains specific knowledge and is generally built for a particular application or domain of discourse [3]. The two structures are complementary in problem solving. Ontologies could even be considered to be hard-coded in computer systems [4] as they represent factual knowledge not varied across applications. A formalization of ontology based on Order-Sorted Logic has also been proposed [1][5], together with an application to upper event ontology [6]. In this logic, an ontology is represented by a "sort hierarchy" and a "predicate hierarchy". The former is a hierarchy of objects in the domain of discourse, built according to a set of partially ordered sorts. In addition, those objects could be further described through their n-ary predicates.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 79–85, 2008.

The relationships between these predicates form another hierarchy, called predicate hierarchy, which complements the sort hierarchy in the ontology.

In this paper, we attempt to enhance the conceptual structure ontology formalism by incorporating new ideas from the above approaches, especially with regard to concept relations (or concept predicates). Our formal definitions of properties of concept types, concepts, relation types and relations bear some similarities with the ontological conceptual ideas proposed by Dillon et al. [3]. The end result is the production of a more complete ontology, that we call the closure of the original ontology, in which missing arguments in relation types are supplemented and their properties propagated. Our main motivation for this research is in the area of formal reasoning, of which one application is the development of systems that can answer queries concerning topics that do not explicitly exist in databases, through automated inference based on ontological relationships between database objects and their predicates. For example, in the justice administration domain, we may wish to have a system that can automatically answer the following question: "Knowing only that John's father is in jail, does John have a Police record and is he being monitored by a welfare agency?" We will see at the end of this paper how an ontology built according to our formalism could help answer this question. Note that Sections 3 and 4 contain new formalizations that have not been previously published.

2 Ontology Formalization in Conceptual Structure Theory

This Section summarizes previous work on ontology formalization within the Conceptual Structure Theory [2][8][9].

Definition 1 (Original Ontology). An ontology $\boldsymbol{O}$ is a 5-tuple $\boldsymbol{O} = (T, I, <, conf, B)$ in which:

(1) T is the *set of types*, i.e., $T = T_C \cup T_R$ where T_C is the set of *concept types* and T_R, the set of *relation types*.
(2) I is the *set of individuals* or *instances* of concept types in T_C.
(3) "<" is the *subsumption relation* in T, representing the semantic generalization or specialization relationship between two types. "<" is mathematically *reflexive.*
(4) *conf* is the *conformity* function that links each individual in I to the infimum (or greatest lower bound) of all concept types that could represent that individual.
(5) B is the *canonical basis* function that defines for each relation type in T_R the tuple of all concept types that can be used in that relation type. For a relation type r, the number of elements in $B(r)$ is called the *arity* (or *valence*) of r or of $B(r)$.
(6) The function B must also satisfy the following association rule:
***B*-rule:** *If a relation type subsumes another relation type, then they must have the same arity and their values through B (i.e., the two tuples of concept types) must also be related (through the subsumption relation <) in the same order.*

Note:

- In Def. 1(5), the function B expresses the *usage pattern* (or *canonical basis*) of each relation type as it identifies all concept types that can be used in that relation type, i.e., B: $T_R \rightarrow \tau(T_C)$ where $\tau(T_C)$ is the set of all tuples over T_C, formally defined as $\tau(T_C) = \cup_{\{n>0\}} (T_C)^n$. Since $B(r)$ is the most important feature for a

relation type *r*, we usually represent *B(r)* together with *r*. For example, if *r=isDaughterOf,* then we often write *r=isDaughterOf(Woman, Person)*, which means that we write *r(B(r))*, and we usually say that *Woman* and *Person* are the two arguments of the relation type *isDaughterOf* while in reality they are the two arguments of the 2-tuple *B(isDaughterOf)*. An example of the *B*-rule (Def. 1(6)) is "*isDaughterOf* < *isChildOf*" with *B(isDaughterOf)=(Woman, Person)* and *B(isChildOf)=(Person, Person)*. In this case, the *B*-rule imposes that the first concept argument in the first relation is subsumed into the first argument of the second relation (i.e., *Woman* < *Person*), and likewise for the second arguments (i.e., *Person* < *Person*).

3 Proposed New Ontology Formalism

We first introduce the new mathematical concepts of tuple extension and tuple subsumption, and then use them in our new ontology formalization.

Definition 2 (Tuple Extension and Tuple Subsumption).

(1) A component c of a tuple T is written as $c \in T$ (normally this notation is reserved to *set* membership). Note that in a tuple the same component may appear multiple times and the order in which the components are written is significant.

(2) Tuple Extension: Let $T_1=<c_1, \ldots, c_n>$ be an n-tuple and $T_2=<d_1, \ldots, d_m>$ be an m-tuple, T_1 is said to be an extension of T_2 (and we write $T_1=\text{ext}(T_2)$) if and only if all components of T_2 are also present in T_1 and in the same order, i.e. $T_1=\text{ext}(T_2) \Leftrightarrow <c_1, \ldots, c_n>= \text{ext}(<d_1, \ldots, d_m>) \Leftrightarrow (m \le n)$ and $(\forall k,l\ 1 \le k \le l \le m\ \exists i,j$ with $1 \le i \le j \le n$ and $c_i = d_k$ and $c_j = d_l)$.

(3) Tuple Subsumption: This is an extension of the definition of the subsumption relation "<" in Def. 1(3). Let T_1 be an n-tuple and T_2 be an m-tuple with $m \le n$, T_1 is said to subsume T_2 (and we write $T_2 < T_1$) if there exists an m-tuple T_2' such that T_1 is an extension of T_2' (i.e., $T_1=\text{ext}(T_2')$) and each component of T_2 is subsumed into the corresponding component of T_2' (i.e., if $T_2 = <c_1, \ldots, c_m>$ and $T_2' = < c_i', \ldots, c_m'>$ then $\forall i\ 1 \le i \le m\ \ c_i < c_i'$).

Property 1 (Tuple Extension). Tuple extension is a relation that is:

- reflexive (i.e., $\forall T\ \ T=\text{ext}(T)$),
- anti-symmetrical (i.e., $\forall T_1, T_2\ \ T_1=\text{ext}(T_2)$ and $T_2=\text{ext}(T_1) \Rightarrow T_1=T_2$), and
- transitive (i.e., $\forall T_1, T_2, T_3\ \ T_1=\text{ext}(T_2)$ and $T_2=\text{ext}(T_3) \Rightarrow T_1=\text{ext}(T_3)$)

Definition 3 (New Ontology with Relation Type Hierarchy). An ontology $\boldsymbol{O}$ is a 5-tuple $\boldsymbol{O} = (T, I, <, conf, B)$ as per Def. 1 with in addition the following features:

(1) The *set of individuals* or *instances I* is expanded to include all instances of concept types and relation types, i.e., $I=I_C \cup I_R$ with I_C being the set of all *concepts* (or instances of concept types) and I_R the set of all *relations* (or instances of relation types).

(2) The function *conf* is extended to be defined over the combined set of I_C and I_R , i.e., $conf: I_C \cup I_R \rightarrow T_C \cup T_R$

(3) The function B is extended to be defined over the combined set of T_R and I_R , i.e., $B: T_R \cup I_R \rightarrow \tau(T_C) \cup \tau(I_C)$

(4) The *B*-rule is broadened as follows:
New *B*-rule (Relation Type Extension). If a relation type *r* subsumes another relation type *r'* (i.e., *r'* < *r*), then it is possible to build a relation type *r^* (called an extension of *r* with respect to *r'*) such that *r'* < *r^*, *B(r^)=ext(B(r))*, *B(r')<B(r^)* and *B(r)<B(r^)*.

Note:

- Def. 3(1) introduces the new concept of instance of relation type to Conceptual Structure Theory. Def. 3(2) means that the new *conf* function associates each concept (resp. relation) with a concept type (resp. relation type), which is the (unique) infimum of all concept types (resp. relation types) in the ontology that the concept (resp. relation) could represent. Def. 3(3) means that in addition to linking a relation type to a tuple of concept types that can be used with that relation type, *B* also links a relation to a tuple of concepts that are currently used in that relation. In plain terms, the new *B*-rule (Def. 3(4)) states that if a relation type *r* subsumes another relation type *r'*, then it is possible to build a relation type *r^* that extends the arguments of *r* such that each argument of *r'* is subsumed into a corresponding argument of *r^*. In other words, the arguments of *r'* (in fact their supertypes) are "merged" into the arguments of *r* to create the set of arguments for *r^*. For example, if we have: *steal(Thief, TheftVictim)* and *offend(Offender)* with *steal* < *offend*, then we could construct the extended relation type: *offend^(Offender, OffenceVictim)*, by adding a supertype of *TheftVictim* (which is *OffenderVictim*) to the tuple of arguments of the new type. The new *B*-rule is the first step to supplement missing arguments in relation types, similarly to the manipulation of predicate arguments in order-sorted logic [5][7].

4 Properties of New Ontology Formalism

The main difference between a relation type and a relation is that the latter may include specific information that is pertinent to the particular context in which the concerned concepts are linked. For example, *isDaughterOf* is a relation type, linking two concept types: *Woman*, *Person*. To express that "Mary is the daughter of John by adoption", we can use the relation type *isDaughterOf* but with a qualifier *byAdoption*. This means that the two instances of the concept types *Woman* and *Person* (which are *Mary* and *John*) are linked through a particular instance of the relation type *isDaughterOf,* which contains the additional qualifier: *byAdoption* (this will be formally defined as a *property* of the relation). And we write: *isDaughterOf(Woman: Mary, Person: John, <byAdoption>)*. In general, a relation contains specific information that is not already contained in the concepts that it links. In the example, the qualifier *byAdoption* is not specific to the concept *Mary*, nor to the concept *John*, but is specific to a particular case (i.e., an instance) of the relation type *isDaughterOf*. If the specific information of the relation can be accommodated by other concept types (that are already in the concept type hierarchy of the ontology), then that specific information should be added to those concept types (rather than as a *property* of the relation). If we have *ChildParentRelationship* as a concept type in the ontology, then we can have a 3-ary relation: *isDaughterOf(Woman: Mary, Person: John, ChildParentRelationship: Adoption)*. The decision to express a piece of information in a relation as a

property or as a new concept type usually depends on the domain of discourse and on practical constraints on that ontology, as the introduction of any new concept type requires a review of existing relations and relation types in the ontology and may lead to proliferation of concept types of minor significance in the ontology.

The propagation of properties between types and instances are governed by the following four axioms:

Axiom 1 (Type Property Inheritance). For any type, its properties are inherited by all of its instances, and by all of its subtypes.

Axiom 2 (Instance Property Generalization). For any instance of a type and for any supertype of that type, one can build another instance of that supertype such that the properties of the first instance also hold true for the second instance.

Axiom 3 (Relation Type Closure). For any relation type *r*, one can build another relation type, called the closure of *r* and written as *r**, satisfying the following conditions:

(1) *r** contains all the arguments of *r,* together with all the properties of *r* and all the properties of the arguments of *r*, if exist.
(2) *r** contains all the arguments of each supertype of *r*, with possibly additional properties for those arguments (i.e., properties that are specific to the semantics of *r*).
(3) For each subtype of *r* and for each argument of that subtype, *r** contains a supertype of that argument, together with all properties of that argument, if exist.
(4) *r** contains no semantically redundant arguments.

Axiom 4 (Relation Closure). For any relation *i* of type *r*, one can build another relation, called the closure of *i* and written as *i**, such that *i** is an instance of the relation type closure *r**. In addition, *i** contains all the arguments of *i*, together with all the properties of *i*, and all the properties of the arguments of *i*, if exist.

Definition 4 (Ontology Closure). For an ontology ***O***, the ontology ***O**** obtained by adding all the relation type closures and relation closures built as per Axioms 3 and 4 is called the closure of the ontology ***O***.

Proposition 1 (Soundness of Relation Type Extension and Closure). For any relation types *r* and *r'* such that *r'* < *r*, *r** is an extension of *r* with respect to *r'* (in the sense of the new *B*-rule).

Note:

- Proposition 1 means that the closure of a relation type is the final result obtained by recursively "extend that relation type with respect to each of its subtypes", i.e., all the arguments of its subtypes are "merged" into the arguments of the relation type to produce its closure. In addition, the closure of a relation type or a relation inherits multiple properties from its supertypes, its subtypes, and its arguments. Semantically, there is no extra information introduced by the notion of closure but the addition of all possible arguments and properties that a relation type or relation could use facilitates inferences and searches on knowledge bases.

Example 1. Suppose that we have the following information in an ontology:

- *pickPocket(PettyLarcenist, PickpocketVictim, StolenAmount)*
- *steal(Thief)*
- *offend(Offender, OffenceVictim, OffenceAct, OffenceInstrument)*

- *pickPocket(PettyLarcenist: John, PickpocketVictim: Mary, StolenAmount: $5.00)* (i.e., "John picked $5.00 from Mary's pocket.").

As per Axioms 3 and 4, we can infer the following relation closures:

- *pickPocket*(PettyLarcenist: John, PickpocketVictim: Mary, OffenceAct: <pickPocket>, OffenceInstrument: <byHands>, StolenAmount: $5.00)*, i.e., "John picked $5.00 from Mary's pocket.".
- *steal*(Thief: John, TheftVictim: Mary, OffenceAct: <pickPocket>, OffenceInstrument: <byHand>, StolenObject: $5.00)*, i.e., "John steals $5.00 from Mary by picking with his hand in Mary's pocket" (this is inferred from the above relation *pickPocket** and from the subsumption relation *pickPocket* < *steal*).
- *offend*(Offender: John, OffenceVictim: Mary, OffenceAct: <pickPocket>, OffenceInstrument: <byHand>, OffenceMotive: $5.00)*, i.e., "John commits an offence against Mary by picking $5.00 with his hand from Mary's pocket" (this is inferred from the above relation *steal** and from the subsumption relation *steal* < *offend*).

Proposition 2 (Soundness of Relation Type Closure). Let r be a relation type. We have: (1) $(r^*)^* = r^*$ (2) $r^* < r$, and (3) Each argument of r^* is the infimum of all the semantically-related arguments of all supertypes of r and of an argument of r, if exists.

Note:

- Proposition 2 expresses that each relation type closure is a semantic specialization of the corresponding relation type and the closure of a relation type incorporates the semantics of the relation type as well as the semantics of the part of the ontology in the background that concerns that relation type (i.e., the relevant concept types that the relation type links, and their supertypes). Therefore, we can say that in an ontology closure, the semantics of a relation type and its context are *condensed* into the relation type closure.

Example 2 (Justice System). Let us suppose that we have the following justice-related information, derived from facts and common findings:

(1) Any offender would have a record with Police.
(2) Children in a dysfunctional family are more likely to offend.
(3) Children in a family whose parents are often absent are monitored by a welfare agency (for possible assistance).

Suppose that we also have in our knowledge database the only piece of information concerning an adolescent named John, that is "John's parents are in jail". We would like to construct a system able to automatically answer the following queries: Is John being monitored by a welfare agency? And does John have a Police record? To answer these questions, we first organize the above general information into an ontology with the following 3 relation types and 1 concept type:

- *hasParentsInJail(Person)*
- *hasAbsentParents(Person, MonitoringWelfareAgency)*
- *isInDysfunctionalFamily(Person, Offence: <moreLikely>)*
- *Offence: <hasPoliceRecord>*

Semantically, we have the following subsumption relations between the above relation types: *hasParentsInJail* < *hasAbsentParents* < *isInDysfunctionalFamily.* From

the information in the knowledge base, we also have the relation: *hasParentsInJail(Person: John).* Based on Axioms 1, 3 and 4, we can deduce the closure of that relation: *hasParentsInJail*(Person: John, MonitoringWelfareAgency, Offence: <moreLikely><hasPoliceRecord>).* From that new relation closure, we can extract the answer to our questions: "John is being monitored by a welfare agency, more likely to offend, and more likely to have a Police record". This answer is possible through the use of the ontology closure in our new formalism.

5 Conclusion

This paper presented an extension to the ontology formalization previously proposed for the conceptual structure theory, by integrating new ideas from order-sorted logic and other logics. The enhanced formalism offers a more rigorous interpretation of the semantic relationships between concepts and their predicates. Unlike OWL, our ontology formalism contains an additional hierarchy constructed by n-ary relation types. The new structure formalizes ontological relationships among any number of concept types. Other new notions introduced in this paper help incorporate predicate properties into the ontology. In particular the new definitions of closures of a relation and of a relation type enable completion of their missing arguments. The end result is the production of an "ontology closure", which is both complete and sound for formal automated reasoning. Based on such an ontology, we could answer queries concerning topics that are not explicitly present in existing knowledge bases.

References

1. Cohn, A.G.: Taxonomic reasoning with many sorted logics. Artificial Intelligence, Rev. 3, 89–128 (1989)
2. Corbett, D.: Reasoning and Unification over Conceptual Graphs. Kluwer Academic Publishers, New York (2003)
3. Dillion, T., et al.: Differentiating Conceptual Modelling from Data Modelling, Knowledge Modelling and Ontology Modelling and a Notation for Ontology Modelling. In: 5th Asia-Pacific Conference on Conceptual Modelling, Wollongong, Australia (January 2008)
4. Greiner, R., Darken, C., Santoso, N.: Efficient Reasoning. ACM Computing Surveys 33(1), 1–30 (2001)
5. Kaneiwa, K.: Order-sorted logic programming with predicate hierarchy. Artificial Intelligence 158(2), 155–188 (2004)
6. Kaneiwa, K., Iwazume, M., Fukuda, K.: An Upper Ontology for Event Classifications and Relations. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 394–403. Springer, Heidelberg (2007)
7. Nitta, K., et al.: New HELIC-II: A software tool for legal reasoning. In: 5th Int. Conf. on Artificial Intelligence and Law, College Park, MD, pp. 287–296. ACM Press, New York (1995)
8. Nguyen, P., Corbett, D.: A Basic Mathematical Framework for Conceptual Graphs. IEEE Transactions on Knowledge and Data Engineering 18(2), 261–271 (2006)
9. Nguyen, P., Corbett, D.: Building Corporate Knowledge through Ontology Integration. In: Hoffmann, A., Kang, B.-h., Richards, D., Tsumoto, S. (eds.) PKAW 2006. LNCS (LNAI), vol. 4303, pp. 223–229. Springer, Heidelberg (2006)
10. World Wide Web Consortium, OWL Web Ontology Language - Use Cases and Requirements (2004), `http://www.w3.org/TR/webont-req/`

Exploiting Ontological Structure for Complex Preference Assembly

Gil Chamiel and Maurice Pagnucco

School of Computer Science and Engineering, The University of New South Wales, NSW, Sydney 2052, Australia and NICTA, Sydney, Australia
{gilc,morri}@cse.unsw.edu.au

Abstract. When a user is looking for a product recommendation they usually lack expert knowledge regarding the items they are looking for. Ontologies on the other hand are crafted by experts and therefore provide a rich source of information for enhancing preferences. In this paper we significantly extend previous work on exploiting ontological information by allowing the user to specify preferences in a more expressive manner. Rather than allowing for only one preferred target concept, we allow a 'chain' of user preferences. Furthermore, we treat information from the underlying ontology of the domain as a secondary preference structure. We then show how to assemble these two preference structures (user and ontology) into a preference over items.

The ability to model preferences and exploit preferential information to assist users in searching for items has become an important issue in artificial intelligence. Accurately eliciting preferences from the user in the form of a query can result in a coarse recommendation mechanism with numerous results returned. In most cases the user lacks deeper, expert knowledge of the domain to allow for a discriminating recommendation to be determined but they know what they like. In the model that we advocate here, user preferences are supplemented and enhanced by expert knowledge in the form of ontologies.

In previous work [1], we show how to exploit ontological information to enhance preferences expressed by a user who is seeking a recommendation for items. In this paper we significantly extend these ideas by allowing the user to specify preferences in a more expressive manner rather than just a single target concept. We allow the user to specify a partial preference over concepts. Moreover, we treat information from the underlying ontology of the domain as a secondary preference structure (i.e., the 'expert knowledge'). We then show how to assemble these two preference structures (user and ontology) into a total preference ordering—and hence recommendation—over items.

Previous work in supplementing user preferences with ontological information is limited. An ontology based similarity system has been presented in [2] but provides for only basic features. [3] is much closer in spirit to this paper, introducing the notion of *ontology filtering*. However, they propose a score propagation system within an hierarchical graph, where we focus on the structural properties of the ontology. [4] has applied similarity to semantic data mapping, ontology mapping and semantic web service matchmaking using techniques from linguistic analysis. [5] deals with preference query relaxation plans over multiple conceptual views.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 86–92, 2008.

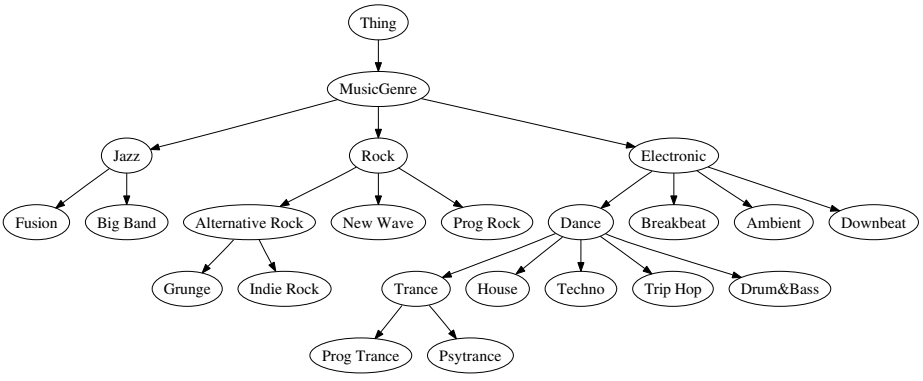


Fig. 1. Ontological Concept Hierarchy of Music Genres

Motivating Example: The Music Record Shop Ontology Consider an ontology which describes and stores information about music albums. This may include musical genre, performing artists, price etc. Let's also consider a concept hierarchy over genres composed by musical experts given in Figure 1. Note that this information is taken from domain experts while we do not assume the user has extensive knowledge of the domain at all. Still this information can be readily exploited when reasoning with preferences. Note that in this hierarchical genre system an album can be identified, for example, as *Progressive Rock* which is a leaf concept but at the same time an album can be identified with the concept *Rock* even though it serves as an abstract concept.

In [1] we discuss ways to exploit this hierarchy in order to perform more accurate preference querying when the preference is given in the form of a single target concept (i.e., the user's most desirable concept classifying the objects over which they seek recommendations). In this paper we significantly extend these ideas by considering complex preference construction.

1 Background

One of the most distinctive properties in representing knowledge using ontologies is the inherent ability to define the terminologies in the ontology in an hierarchical manner. In analogue to terminologies in database systems, i.e. the database schema, in this work we consider the terminological component of an ontology to be the part which stores information created by domain experts. This information holds in all circumstances. This assumption will then enable us to enhance the user expressed preferences with information defined by domain experts, assuming limited domain knowledge on behalf of the user, and thus supply more accurate preference querying. In [1] we introduce an extension to [6] by allowing the user to express their preferences in terms of either the hierarchy below the target concept or the similarity to a certain target concept. In [1] we explore various similarity metrics to assist in preference querying as described above. We introduce a new Boolean operator $Sim(C_1, C_2)$ that can be read "is similar to" and

returns a real number in the interval $(0, 1]$ that specifies the degree of similarity between concepts C_1 and C_2:

$$b_1 \prec_{P(C_0)} b_2 \Leftrightarrow Sim(C(b_1), C_0) < Sim(C(b_2), C_0) \quad (1)$$

Where $C_0 \in Concepts$ is the target concept, $b_1, b_2 \in ResultBindings$ (i.e., the objects over which we seek recommendation) and $C(b_i)$ is the value bound to the relevant variable in the result binding b_i w.r.t C. In [1] we propose a novel similarity method, based on [7], which has three interesting properties:

1. It considers two concepts more similar if they share more specific information.
2. It respects the *IS-A* (subsumption) relation.
3. Within a sub-graph, it will consider a concept more similar to a target concept according to the communicated level of specificity described by this target concept.

Property 2 means that when the user specifies a certain target (most preferred) concept, the sub-concepts below this target concept will be always considered more similar to the target concept than any concept which is not subsumed by this target concept.

Property 3 means that those sub-concepts below the target concept will be ordered according to their distance to the target concept (the 'closer' distance, the more similar they are). The intuition behind this is to respect the user's communicated level of specificity (given by the depth of this target concept in the ontology) considering concepts which are 'closer' to this level of specificity to be more similar. This is measured by the following similarity metric which determines the similarity of concepts C_0 and C_1.

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG} \quad (2)$$

Where N_1, N_2 are the distances from the concepts C_0 and C_1 to their Most Recent Common Ancestor (MRCA) respectively and N_3 is the distance from this MRCA and the root of the ontology (assuming the most general concept is the OWL concept *Thing*). AVG is the average distance of MAX to the depth of the concepts C_0 and C_1 and MAX is the length of the longest path from the root of the ontology to any of its leaf concepts. Note that for clarity this figure is then being normalized to be between 0 and 1 by dividing it by the similarity of the target concept to itself.

Example 1. *Suppose we query music albums preferring albums of genre* ***similar to*** Alternative Rock. *An album classified as* Alternative Rock *will be considered the most similarly matched to our preference with similarity of 1. The genre* IndieRock *will have similarity of* ~ 0.94 *while the genre* Jazz *will have the similarity of* ~ 0.35.

While this technique adds some flexibility to preferential reasoning, it provides a pathway to a much richer way of reasoning about preferences that we explore in this paper.

2 Complex Preference Assembly Using Ontological Structure

In [1], complex preference assembly is done through the Pareto (all equal) and the Cascade (prioritization) operators. In particular, the current semantics of preference prioritization becomes less effective and somewhat limiting when dealing with similarity-based preferences. The reason is inherent in the way Cascade performs: given two preference

constructors over the same attribute, the second preference will only come into consideration for those cases where the two alternatives are considered equally preferred by the first preference. In similarity-based preference querying we allow the system to consider alternatives which are 'close' to the expressed preference although not necessarily the exact ones expressed by the user. And thus, Cascade will consider a second alternative only for those result bindings which have the exact same similarity measurement to the first concept. We remedy this problem by introducing a new preference assembly method: *PreferenceOrdering* which allows the user to give a partial preorder of preferences over concepts. The system then completes this order using ontological information. Our assumption here is that the user is rarely a domain expert. The user knows what they like and these wishes must be respected. It is more than likely however that the preferences expressed by the user are not based on any deeper insight about the domain. We exploit ontological information about the domain in order to provide a much more discriminating recommendation for items. More formally, the user provides a preference over items represented as a partial preorder over concept classes:

$$\{C_{1,1}, C_{1,2}, \cdots, C_{1,i}\} > \{C_{2,1}, C_{2,2}, \cdots, C_{2,j}\} > \cdots > \{C_{n,1}, C_{n,2}, \cdots, C_{n,k}\}$$

Each set $\{C_{i,1}, C_{i,2}, \cdots, C_{i,n}\}$ represents concepts that the user equally prefers. We only require that preferences be consistent; i.e., transitivity and asymmetry preserved. *Our goal now is to turn this into a total order by "filling out" user preferences with information from the ontology by utilising notions of similarity. We end up with a total preorder $C_1 > \cdots > C_m$ over all concepts in the ontology with those corresponding to concepts specified in the user preference maintaining the order imposed by the user.* By doing so we provide a much more fine-grained recommendation for the user that is less likely to overwhelm them with choices. Note that we do allow indifference between two items as well.

We now present a method for assembling a preference ordering by using the notion of similarity between concepts measured in terms of the structure of the ontology. We first give some basic definitions:

Definition 1. *Given a sub-graph of an ontology, a* User Ordering *is a partial preorder over the concepts of that sub-graph given by the user.*

Definition 2. *Given a sub-graph of an ontology (a conceptual view) and a preferred target concept, an* Ontology Ordering$_{Sim}$ *is the total order of concepts in that sub-graph according to the similarity method* Sim.

In order to assemble the two different orders (User and Ontology), we create a total order over the relevant part of the graph. The position of any of these concepts in the total order will be determined by looking at their maximal similarity to any of the user ordered concepts. Concepts which are most similar to a user ordering concept will be then ordered according to their similarity to it. The intuition behind this is that we exploit the ordering given to us by the ontology (w.r.t a similarity measurement) without contradicting the preference ordering explicitly expressed by the user. Suppose the user specifies their preference using the following informal syntax:

$$(C_{1,1}...C_{1,i}) \quad \texttt{THEN} \quad (C_{2,1}...C_{2,j}) \quad \texttt{THEN} \quad \cdots \quad \texttt{THEN} \quad (C_{n,1}...C_{n,k})$$

Given a similarity method $Sim(C_0, C_1) \in (0, 1]$, the semantics of the ordering of preferences (w.r.t a candidate concept C_0) is:

$$b_1 \prec_{P(C_1,C_2,\cdots,C_n)} b_2 \Leftrightarrow Sim(C(b_1), \langle C_1, C_2, \cdots, C_n\rangle) < Sim(C(b_2), \langle C_1, C_2, \cdots, C_n\rangle) \tag{3}$$

Where

$$Sim(C_0, \langle C_1, \cdots, C_n\rangle) = \left\{ Sim(C_0, C_i) \times \prod_{j=1}^{i} Sim(C_j, C_i) | i = argmax_{k=1}^{n} Sim(C_0, C_k) \right\} \tag{4}$$

We allow C_i to be a primitive concept or a set of equally preferred concepts. To take care of the latter $\{C_{i,1}, C_{i,2}, \cdots, C_{i,m}\}$ we look at the maximum similarity between C_0 and any concept in the set. In general, $Sim(\{C_1, \cdots, C_n\}, \{C_1, \cdots, C_m\})$ is determined by the maximal similarity between any concept in the first set and any concept in the second.

In formula 4, the user's preferences take precedence. Concepts in the underlying ontology that are not specified in the user's preferences are used to "fill out" this ordering. Intuitively, each such concept is ordered after the concept in the user's preference ordering to which it is most similar according to the adopted similarity measure. Furthermore, these concepts are ordered among themselves according to the strength of this similarity.

Example 2. *The user specifles the following partial preorder:*

$$\{\textit{AlternativeRock, ProgRock}\} \texttt{ THEN } \{\textit{Electronic}\}$$

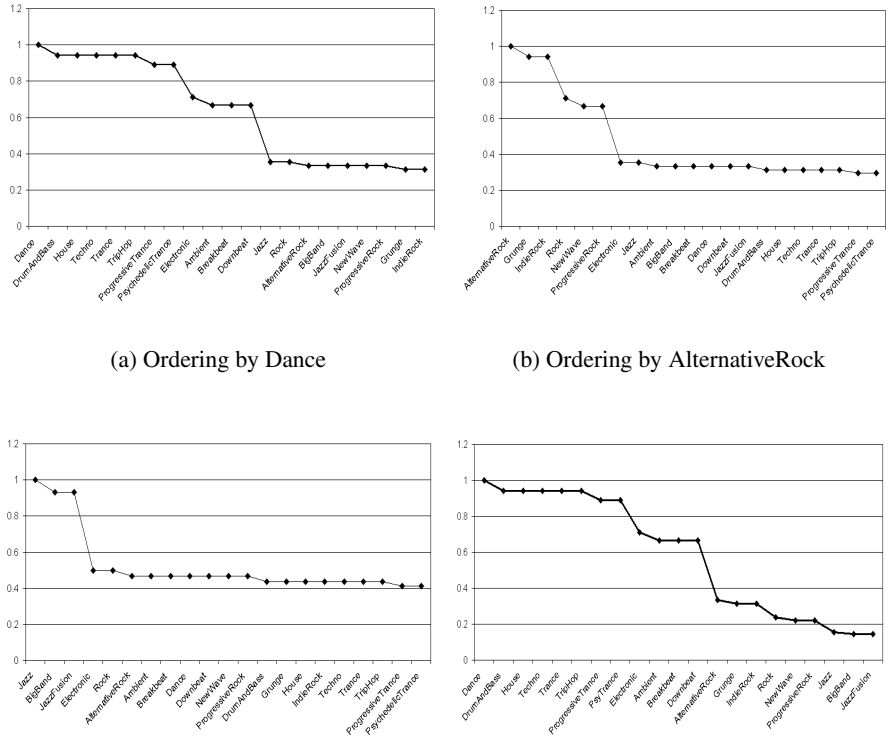
Given a similarity method with the semantics of (2), the total order created by (4) is:

$$\{\textit{AlternativeRock,ProgRock}\} > \{\textit{Grunge,IndieRock}\} > \{\textit{other rock concepts}\} > \{\textit{Electronic}\} > \{\textit{Ambient,Breakbeat,Dance,Downbeat}\} > \{\textit{other Electronic concepts}\} > \{\textit{Jazz concepts}\}$$

AlternativeRock *and* ProgRock *perfectly match the first preference and appear at the top of the total order. Concepts are then ordered according to their similarity to these target concepts until we reach a concept more similar to the second preference: this is the concept* Electronic *(which perfectly matches the second preference). Concepts are then ordered according to their similarity to* Electronic *to fill in the total order.*

3 Analysis

We evaluate our preference ordering assembly method by looking at some intuitive examples and examining the behaviour of the resulting total orders. We base our analysis on the concept similarity method defined in (2). Let us have a closer look at this operator. In order to preserve the properties listed in Section 1, this operator was designed so it will have different behaviours when comparing a target concept with a descendant to



(a) Ordering by Dance

(b) Ordering by AlternativeRock

(c) Ordering by Jazz

(d) Assembled Preference Ordering

Fig. 2. Similiarity-based Preference Ordering Analysis

when comparing a target concept to an ancestor. It can be deduced that if $C_1 \sqsubseteq C_0$ and $C_2 \sqsubseteq C_1$ then

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG}$$

$$Sim(C_2, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG + \frac{1}{2}\Delta}$$

Where Δ is the direct distance between C_1 and C_2. And in the case where $C_0 \sqsubseteq C_1$ and $C_1 \sqsubseteq C_2$:

$$Sim(C_2, C_0) = \frac{2 * N_3 + 2\Delta}{N_1 + N_2 + 2 * N_3 + AVG - 1\frac{1}{2}\Delta}$$

So, for example, when assigning the target concept to be *Dance* (Figure 2(a)) we see that the similarity measurement linearly decreases for descendant concepts w.r.t *Dance* while every 'climb' up to an ancestor reduces the similarity measurement significantly (and non-linearly). Intuitively, you can say that the difference in similarity when comparing elements 'in the area of' Dance is bigger than the difference in similarity between

elements which are quite 'far' from this target concept and thus the ordering according to it is less relevant. These 'steps' actually come in handy when assembling the preference ordering:

Example 3. *Consider the following query ordering:* Dance *THEN* Alternative Rock *THEN* Jazz*. Figures 2(a)–2(c) give us the different orderings for each preference level while Figure 2(d) shows the assembled preference ordering. Being the first preference,* Dance*, its descendants (its 'step') along with the following step which (according to this similarity method) is considered similar enough to Dance are ordered according to it (as in Figure 2(a)). The step of the second preference is ordered according (as in Figure 2(b)) to it where this sub-order is no longer influenced by the first preference and like-wise for the third preference.*

In other words, the preference ordering assembly identifies which user preference constructor is to be ordered upon and does that without contradicting the user preferences.

4 Conclusions

We have presented a novel technique for augmenting a partially specified user preference by structural information in a domain specific ontology. In particular we 'complete' the user specified preference ordering by using expert knowledge in the form of structural properties of an ontology to determine how concepts not explicitly ordered by the user should be assembled and ordered with respect to preferences explicitly supplied by the user. We present a method for determining this ordering based on a (quantitative) similarity measure. Furthermore, we have analyzed our approach through a number of illustrative examples. It is easy to see that this form of preferential reasoning provides an expressive mechanism for naive and non-expert users to specify preferences that can be turned into a discriminating recommendation for them by using the structure of the underlying ontology.

References

1. Chamiel, G., Pagnucco, M.: Exploiting ontological information for reasoning with preferences. In: Proc. 4th Multidisciplinary Workshop on Advances in Preference Handling (2008)
2. Middleton, S.E., Shadbolt, N.R., De Roure, D.C.: Ontological user profiling in recommender systems. ACM Trans. Inf. Syst. 22(1), 54–88 (2004)
3. Schickel-Zuber, V., Faltings, B.: Inferring User's Preferences using Ontologies. In: AAAI 2006, pp. 1413–1418 (2006)
4. Kiefer, C., Bernstein, A., Stocker, M.: The fundamentals of iSPARQL – A virtual triple approach for similarity-based semantic web tasks. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825. Springer, Heidelberg (2007)
5. Balke, W.T., Wagner, M.: Through different eyes: assessing multiple conceptual views for querying web services. In: Proc. of the 13th international World Wide Web, pp. 196–205 (2004)
6. Siberski, W., Pan, J.Z., Thaden, U.: Querying the semantic web with preferences. In: International Semantic Web Conference, pp. 612–624 (2006)
7. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: 32nd Annual Meeting of the Association for Computational Linguistics, pp. 133–138 (1994)

A Refutation Approach to Neighborhood Interchangeability in CSPs

Chavalit Likitvivatanavong and Roland H.C. Yap

School of Computing, National University of Singapore
{chavalit,ryap}@comp.nus.edu.sg

Abstract. The concept of Interchangeability was developed to deal with redundancy of values in the same domain. Conventional algorithms for detecting Neighborhood Interchangeability work by gradually establishing relationships between values from scratch. We propose the opposite strategy: start by assuming everything is interchangeable and disprove certain relations as more information arises. Our refutation-based algorithms have much better lower bounds whereas the lower bound and the upper bound of the traditional algorithms are asymptotically identical.

1 Introduction

Interchangeability was introduced in [1] in order to deal with redundancy of values in the same variable domain. Removing or grouping interchangeable values together has proved useful in reducing search space and solving time [2–5].

Neighborhood Interchangeability (NI) can be detected in quadratic time by the Discrimination Tree algorithm (DT) [1]. DT works by assuming zero knowledge and build up relationships between values. The disadvantage is that determining whether a value is NI with another one requires all values to be checked.

We propose a different method that is able to detect values that are not NI early on, without checking all the values. Initially we assume values are identical. That is, they are NI with each other. For each value in the neighboring variables, we test its consistency against these values and update our assumption about their relationships. When enough is known so that a value is certain not to be NI with any other value, it can be removed from future consideration.

In this paper, we will study algorithms that can efficiently identify neighborhood interchangeability using this approach. The paper is organized as follows. Section 2 gives the background for CSPs and Interchangeability. Section 3 gives the overall flow of the algorithms, with concrete algorithms for identifying NI provided in Section 5. We show that these algorithms have much better lower bounds than DT, which is explained in Section 4. We conclude in Section 7.

2 Preliminaries

A finite constraint network P is a pair $(\mathcal{X}, \mathcal{C})$ where $\mathcal{X}$ is a finite set of n variables and $\mathcal{C}$ a finite set of e constraints. Each variable $X \in \mathcal{X}$ has an associated

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 93–103, 2008.

domain containing the set of values allowed for X. The initial domain of X is denoted by $dom^{init}(X)$; the current one by $dom(X)$. Each constraint $C \in \mathcal{C}$ involves an ordered subset of variables of $\mathcal{X}$ called scope (denoted by $scp(C)$), and an associated relation (denoted by $rel(C)$). For each k-ary constraint C with $scp(C) = \{X_1, \ldots, X_k\}$, $rel(C) \subseteq \prod_{i=1}^{k} dom^{init}(X_i)$. For any $t = (a_1, \ldots, a_k)$ of $rel(C)$, called a tuple, $t[X_i]$ denotes a_i. A constraint's relation can be described by a formula (called *intensional form*) or by exhaustively listing all the tuples (called *extensional form* or *positive table constraint*). Alternately, a relation may describe tuples *not* allowed; if the relation is in extensional form it is also called *negative table constraint*. The maximum number of tuples in the network is denoted by s. We denote the maximum size of a domain by d, the maximum arity of all constraints by r and the maximum number of constraints involving a single variable with g.

Let C be a k-ary constraint and $scp(C) = \{X_1, \ldots, X_k\}$, a k-tuple t of $\prod_{i=1}^{k} dom^{init}(X_i)$ is said to be: (1) *allowed* by C if and only if $t \in rel(C)$, (2) *valid* if and only if $\forall X_i \in scp(C)$, $t[X_i] \in dom(X_i)$, (3) *a support* in C if and only if it is allowed by C and valid, and (4) *a conflict* if and only if it is not allowed by C and valid. A tuple t is a *support of* (X_i, a) in C when t is a support in C and $t[X_i] = a$. A *constraint check* determines if a tuple is allowed. A *validity check* determines if a tuple is valid. A *solution* to a constraint network is an assignment of values to all the variables such that all the constraints are satisfied. A constraint network is *satisfiable* if it has at least one solution. A Constraint Satisfaction Problem (CSP) involves determining whether a given constraint network is satisfiable.

We assume that values in different domains are different, so that $a \in dom(X)$ and $a \in dom(Y)$ are different values. The domain must be mentioned to distinguish which a is referred to, unless it is clear from the context.

Some of the Interchangeability concepts introduced in [1] are reviewed below.

Definition 1 (FI). *A value $a \in dom(X)$ is* fully interchangeable *with a value $b \in dom(X)$ if and only if (1) every solution which contains a remains a solution when a is replaced with b, and (2) every solution which contains b remains a solution when b is replaced with a.*

Since identifying FI values amounts to finding all solutions to a constraint network, the process is intractable because the general CSP itself is NP-complete. A weaker but sufficient condition for FI is Neighborhood Interchangeability (NI).

Definition 2 (NI). *Two values $a, b \in dom(X)$ are* neighborhood interchangeable *if and only if for every constraint C such that $X \in scp(C)$,*

$$\{t \in \bar{D} \mid (a, t) \text{ satisfies } C\} = \{t \in \bar{D} \mid (b, t) \text{ satisfies } C\}$$

where $\bar{D} = \prod_{Y \in scp(C) \setminus \{X\}} dom(Y)$.

3 Overall Process

We give general algorithms for identifying and eliminating neighborhood redundant values, Algorithm 1 and 2. A value is redundant if its removal from the

satisfiable constraint network does not render it unsatisfiable. Algorithm 1 can be instantiated appropriately to get the different specific algorithms.

The routine BUILDSTRUCT creates a data structure that allows FILTERSTRUCT to eliminate redundant values efficiently. If FILTERSTRUCT has detected and removed some redundant value, it puts the remaining domain in the data structure filStruct and propagates the result. BUILDDOM rebuilds the domain.

CREATETUPLELIST collects the tuples in the neighborhood. In the worst case, the running time of CREATETUPLELIST as well as the size of tupleList is $O(gd^{r-1})$. If constraints are in extensional form, the running time becomes $O(gs)$, whereas the size of tupleList becomes $O(g.\min(s, d^{r-1}))$. In the rest of the paper we will use l to denote the size of tupleList.

Algorithm 1. REDUNDANCYCHECK($\mathcal{X}, \mathcal{C}$)

```
1 Q ← {X | X ∈ 𝒳};
2 while Q ≠ ∅ do
3     extract X from Q;
4     tupleList ← CreateTupleList(X);
5     valStruct ← BuildStruct(X, tupleList);
6     if FilterStruct(valStruct, filStruct) then
7         dom(X) ← BuildDom(X, filStruct);
8         for C ∈ 𝒞 such that X ∈ scp(𝒞) do
9             Q ← Q ∪ scp(C) \ {X};
```

Algorithm 2. CREATETUPLELIST(X)

```
1 tupleList ← ∅;
2 for C ∈ 𝒞 such that X ∈ scp(𝒞) do
3     if rel(C) is in intensional form then
4         tupleList ← tupleList ∪ ∏_{Y ∈ scp(C) \ X} dom(Y);
5     else
6         for tuple t ∈ rel(C) do
7             t ← the resulting tuple after removing t[X] from t;
8             tupleList ← tupleList ∪ {t};
9 return tupleList ;
```

4 Identifying NI by Discrimination Tree

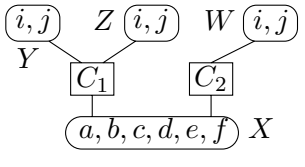
An efficient algorithm for detecting NI values, called the Discrimination Tree algorithm, was introduced in [1]. The idea is to focus on a single value $v \in dom(X)$ and go through values (or tuples for non-binary constraints) in the neighborhood in some fixed order and build a tree based on their consistency

with v. Subsequent checks for other values in $dom(X)$ would begin from the root of the existing tree and follow the path in the tree having the same consistency until reaching the node where consistency differs, at which point a new branch is created. After the algorithm is finished, values that are NI will be grouped in the same leaf. The Discrimination Tree algorithm is shown in Algorithm 3. We remark that the algorithm here is different from the ones in [1, 3] but is still based on a discrimination tree with the same time complexity.

Algorithm 3. BUILDSTRUCT<DT>(X, tupleList)

```
1 create a root node;
2 for a ∈ dom(X) do
3     move to the root node;
4     for t ∈ tupleList do
5         if a is consistent with t then
6             if node corresponding with t existed then
7                 move to that node
8             else construct node corresponding with t
```

For an example of DT, consider the constraint network in Figure 1(i). The tree of this network is shown in Figure 2(i). The algorithm requires 24 constraint checks and reports $\{c, e\}$ as the only set of NI values.



(i)

C_1		
X	Y	Z
a	i	j
b	i	i
b	i	j
c	i	i
e	i	i

C_2	
X	W
c	i
c	j
d	j
e	i
e	j

C_1		
X	Y	Z
b	i	i
c	i	i
e	i	i
a	i	j
b	i	j

C_2	
X	W
c	i
e	i
c	j
d	j
e	j

(ii)

Fig. 1. (i) NI example. Positive tables. (ii) C_1 and C_2 when sorted on (Y, Z) and W.

The worst-case running time of DT is $O(dl)$. We emphasize that the lower bound of DT is $\Omega(dl)$, leaving the algorithm with no room for improvement.

5 Identifying NI by Refutation

We propose an opposite approach to DT in identifying NI values. Instead of starting with zero knowledge about value interchangeability, we assume in the beginning that all values are NI and update the assumption as more data becomes available. We call this algorithm Refutation Tree (RT). Details are shown in Algorithm 4.

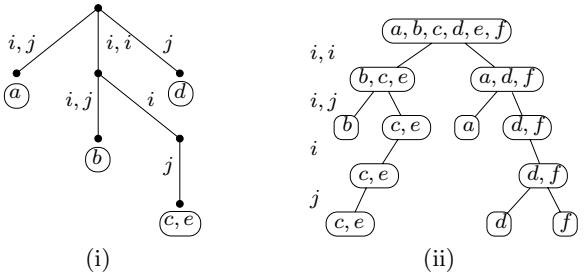


Fig. 2. Two approaches to establish NI: (i) Discrimination Tree (ii) Refutation Tree

Algorithm 4. BUILDSTRUCT<RT>(X, tupleList)

```
1  thisC ← {{v ∈ dom(X)}};
2  for t ∈ tupleList do
3      nxtC ← ∅;
4      for T ∈ thisC do
5          lft ← ∅ ;
6          rgt ← ∅;
7          for a ∈ T do
8              if a is consistent with t then
9                  lft ← lft ∪ {a};
10             else rgt ← rgt ∪ {a};
11         if |lft| ≥ 2 then nxtC ← nxtC ∪ {lft};
12         if |rgt| ≥ 2 then nxtC ← nxtC ∪ {rgt};
13         if nxtC = ∅ then return ∅;
14         else thisC ← nxtC ;
15 return thisC ;
```

We describe the algorithm as follows. The set thisC consists of sets of values, which correspond to the nodes in the refutation tree. The RT algorithm works by traversing the refutation tree in a breath-first fashion. For each tuple in tupleList, the algorithm checks whether it is compatible with the values from each set in thisC (line 4, 7, 8). A set is split into lft set (consistent values) and rgt set (inconsistent values) for each tuple checked. The result represents the current state of knowledge about NI. A singleton indicates that the value in this set is different from the rest of the domain. The value is discarded (line 11 and 12) since no further data would conflict with what we have learned so far. That is, once it is known that v is inconsistent with t, we will not find out later that v is consistent with t.

For example, consider the tree in Figure 2(ii). In the beginning, all values are assumed to be NI. We then check whether they in fact are consistent with the first tuple (i, i) according to C_1 (a value $x \in dom(X)$ is consistent with (i, i) according to C_1 iff $(x, i, i) \in rel(C_1)$). Only b, c, and e are consistent so

Algorithm 5. FilterStruct<RT>(V,F)

```
1 revise ← false ;
2 result ← ∅;
3 for T ∈ V do
4     if |T| > 1 then
5         revise ← true ;
6         pick v ∈ T;
7         result ← result ∪ (T \{v});
8 F ← result ;
9 return revise ;
```

Algorithm 6. BuildDom<RT>(X,F)

```
1 return dom(X) \ F;
```

they are split off to form a new set. We repeat the process with the next tuple until tupleList is exhausted. It takes 20 constraint checks for RT on this example, compared to 24 for DT.

The worst-case time complexity for RT is the same as that for DT. However, RT improves upon DT in the lower bound time complexity. Because each set in the collection thisC is partitioned into at most two sets, the trace of this process forms a binary tree. The lower bound is achieved when the height of the tree is the smallest possible — that is, when the tree is a complete binary tree. Hence, the lower bound is $\Omega(d.\min(l, \lg d))$.

The efficiency of the RT algorithm depends on the number of NI values and the variable ordering when refutation trees are created. In contrast, the cost of DT is fixed regardless of the number of NI values in a domain. RT requires equal or fewer number of constraint checks than DT in all cases.

5.1 Exploiting Table Constraints

The table constraint, when sorted, is recognized as a simple yet effective way to reduce the cost of CSP algorithms [6, 7]. We will show how to exploit sorted table constraints in RT.

Consider the example in Figure 1(i). For tuple (i, i), Algorithm 4 must check the tuple against each value in $\{a, b, c, d, e, f\}$ to partition the set. If the constraints are sorted as shown in Figure 1(ii), this is a simple matter of traversing the table from the first row that contains (i, i) to the first row that contains tuple other than (i, i) and collects values in the X columns while traversing. Once (i, j) is encountered, it is clear that no other values except $\{b, c, e\}$ are consistent with (i, i) so there is no need to check the rest of the table.

The running time for RT with sorted table constraints is $O(s)$. Algorithm 4, however, only provides a high-level concept of the refutation approach but does not say how to exploit table constraints. Since we do not know in advance which values would be encountered during the traversal of tables for a given tuple, this

requires searching in the collection of sets (thisC) for the right sets that contain the same values as in the corresponding section of the table. The cost incurred for the search makes the running time asymptotically higher than $O(s)$.

It is not a trivial task to revise RT so that it is able to exploit sorted table constraints while maintaining its lower bound. In the next section we will give a new algorithm that can process values in any arbitrary order.

Note that DT can also exploit sorted table constraints. In fact, Figure 2(i) can be created just by traversing the tables in Figure 1(i). We will show later that RT can exploit mixed positive and negative table constraints so that the running time can be decreased to even less than $O(s)$.

5.2 Implementation

We provide a detailed algorithm (called RTS) in Algorithm 7. It follows the idea laid out in Algorithm 4 but differs in that it iterates through values in a domain rather than through sets of values in a collection. This is done to facilitate arbitrary orders of values in sorted table constraints. We will explain how sorted table constraints are used in the algorithm later in this section.

We use array bin to partition NI values. Partitions are numbered from 0 to curSize−1. Initially, all values are assigned to bin[0]. For each tuple in the tupleList the algorithm checks whether it is consistent with values in valList. If a value is consistent, it is taken off the current bin and put into a new bin, whose number is determined by array split. Afterward, the size of the original bin (denoted by array size) and the size of the new bin is updated. The value of split is obtained from a linked-list of available bins (nxtOf and next) and is fixed for a given bin until the next tuple is considered (if-block in line 22).

When all values in the same bin are consistent with the tuple t, they are moved to the new bin, leaving the original bin empty. To avoid having the number of bins exceeds the number of values in the domain, we reuse the empty bin and put it in the linked-list of available bins (if-block in line 26).

If the size of a bin is exactly one, the singleton value will be removed from the valList. This is done in the if-block in line 16. We reduce the size of valList by one and swap the singleton value with the value at the end of valList (line 19). We use array label as another abstract layer of values for this purpose.

We use array rec to keep track of the bin assigned involving a given tuple. It serves two purposes. First, it prevents incorrect reuse of bin. Because the value of split[bin[v]] is correct only for a given t (line 8), as soon as a new tuple is considered, the old value of split[bin[v]] is incorrect, since the bin involving the previous tuple become a separate and independent bin. We enforce this condition by comparing the current tuple with the tuple recorded (line 22). Second, we use rec to prevents premature elimination of a singleton value. To be certain that a partition with a single value will not increase in size later, the algorithm must already finish checking all the values against the current tuple. We enforce this condition by eliminating singleton partitions only after the next tuple is considered (line 16).

Algorithm 7. BUILDSTRUCT<RTS>(X, tupleList)

```
1  for i ← 0 to |dom(X)| − 1 do
2      size[i] ← 0 ;      bin[i] ← 0;
3      rec[i] ← ∅ ;     split[i] ← 0;
4      label[i] ← i ;    nxtOf[i] ← i + 1 ;    del[i] ← false ;
5  size[0] ← |dom(X)| ;
6  curSize ← |dom(X)| ;
7  next ← 1;
8  while curSize > 0 and tupleList ≠ ∅ do
9      extract t from tupleList ;
10     if sortedT then valList ← CreateValList(X, t);
11                 else valList ← {0, . . . , curSize − 1};
12     while valList ≠ ∅ do
13         extract i from valList ;
14         if sortedT and del[i] then continue;
15         v ← label[i];
16         if size[bin[v]]=1 ∧ rec[bin[v]]≠t then
17             curSize ← curSize − 1;
18             if sortedT then del[v] ← true ;
19                        else label[i] ↔ label[curSize − 1];
20         else
21             if sortedT or v is consistent with t then
22                 if split[bin[v]]= ∅ or rec[bin[v]]≠t then
23                     split[bin[v]] ← next ;
24                     next ← nxtOf[next];
25                 size[bin[v]] ← size[bin[v]] −1;
26                 if size[bin[v]] = 0 then
27                     nxtOf[bin[v]] ← next ;
28                     next ← bin[v];
29                 rec[bin[v]] ← t;
30                 bin[v] ← split[bin[v]];
31                 size[bin[v]] ← size[bin[v]] +1;
32                 rec[bin[v]] ← t;
```

Example. Let us reconsider the example in Figure 1(i) and the tree in Figure 2(ii). The first tuple is (i, i). After it is checked against values from 0 to 5 (representing $\{a, b, c, d, e, f\}$), we have: split[0]=1, bin[1]=1, bin[2]=1, bin[4]=1 (i.e. bin[1] for b, c, e), bin[0]=0 bin[3]=0 bin[5]=0 (i.e. bin[0] for a, d, f). After (i, j) is checked we have split[0]=2, split[1]=3, bin[0]=2 (for a), bin[1]=3 (for b), bin[2]=1 and bin[4]=1 (for c and e), bin[3]=0 and bin[5]=0 (for d and f). The linked list of available bin (indicated by next together with nxtOf) is 4→5→6. After tuple i is checked, two singletons a and b are removed, and we have curSize=4 and label[0]=5, label[5]=0, label[1]=4, label[4]=1, while the rest of label

remains unchanged. For values 0 to 3 in valList, bin[label[0]]=bin[5]=0 (for f), bin[label[1]]=bin[4]=4 (for e), bin[label[2]]=bin[2]=4 (for c), bin[label[3]]=bin[3]=0 (for d). The linked list of available bin is 1→5→6 (bin 1 and 5 were previously used for a and b, but are available for reuse now that a and b are ignored). □

Sorted table constraints can be exploited by setting flag sortedT to true. We assume the sorted tables are positive. Negative tables can be accommodated simply by changing "consistent" in line 21 to "inconsistent". The algorithm for sorted table constraints differs from the general algorithm in three places. First, valList is set by routine CreateValList, which traverses the corresponding sorted table and collects the relevant values. For instance, using example in Figure 1, valList for (i, i) is $\{b, c, e\}$. Second, we do not perform constraint check for values gathered in this way from sorted table constraints because we already know their consistency just from their existence in the table (line 21). Third, singleton values are skipped in a different way. The same technique for general constraints cannot be used because valList changes from one tuple to another. Instead, we use boolean array del to indicate whether a value should be ignored (line 14 and 18). Notice that this does not decrease the complexity of the algorithm because all the values in valList must be iterated anyway. The lower bound of RTS is $\Omega(s)$.

It is interesting to note that the algorithm is applicable to mixed constraint of both consistency types. The lower bound of $\Omega(s)$ can be made lower if the shorter sections in either positive or negative relations are combined. For instance, suppose we have $dom(X)$=$dom(Y)$=$\{1, 2, 3, 4\}$. $rel(C_1) = \{(1, 1), (1, 2), (2, 2), (3, 2), (4, 2), (2, 3), (4, 3)\}$ lists compatible tuples for X and Y, $rel(C_2)$=$\{(2, 1), (3, 1), (4, 1), (1, 3), (3, 3), (1, 4), (2, 4), (3, 4), (4, 4)\}$ lists incompatible tuples for X and Y. We can gather shorter parts from both constraints to create a mixed constraint C_3, $rel(C_3)$=$\{P(1, 1), N(1, 3), N(3, 3)\}$, where P and N denotes positive and negative tuples. The size of the combined constraint can be much smaller than the size of the original constraints; for this example, $|rel(C_1)| = 7$, $|rel(C_2)| = 9$, $|rel(C_3)| = 3$. We can use C_3 in the algorithm by switching the consistency in line 21 depending on the tuple. While it is rare to have both positive and negative tables for the same constraint, the mixed constraint can be created from only one of them: if the size of the section is less than half the number of all possible tuples then we retain the tuples. If the size is more than half, the tuples in the opposite consistency type would be created by inference.

6 Related Work

NI has been shown to improve search in a number of works [2, 3, 5, 8]. Although DT was introduced only in the context of binary CSPs, it has been extended to cover non-binary CSPs in [3]. The authors pointed to a case where DT provides incorrect results for non-binary CSPs. To avoid this problem, they suggest performing DT on each neighboring constraint and intersecting the results. Our DT is derived from the binary version in a slightly different way and it does not cause the incorrect results.

In [3], it has also been recognized that singleton partitions can be removed. However, these singletons can be eliminated only after DT is finished for each constraint. This makes the lower bound higher than that of refutation-based algorithms, which can ignore singletons much earlier.

7 Conclusion and Future Work

We introduce a refutation method to local interchangeability and study it in detail for NI. Rather than starting with no prior knowledge, we assume the opposite — that everything is interchangeable with one another — and update our assumption as new information comes along. While the algorithms presented have the same upper bound on their running time as those of the standard algorithms, the refutation approach allows some values that are not NI with others to be detected early and removed from further consideration by the algorithms, thus decreasing their lower bounds. We also show how these algorithms can take advantage of table constraints while still achieving the lower bound described.

Note that Algorithm 7 is given with more low-level details because it is not clear how one can take advantage of sorted constraints while maintaining the same lower bound of the generic RT. Without proper care the lower bound could increase, defeating the whole purpose. Otherwise, implementing RT is straightforward.

Another form of local interchangeability called Neighborhood Substitutability [1] also benefits from the refutation approach. NS has more pruning power but received much less attention than NI due to its higher cost. A direct NS algorithm was given in [9] but no experimental result was reported.

Our main objective for this paper is to explore theoretical possibilities that come with this new approach. While the better lower bound does not imply actual performance, it gives practitioners considerable room to maneuver. The practicality of this approach will largely depend on how the average running time can be pushed closer to the lower bound. Since the actual performance of these algorithms are affected by ordering heuristics for the list of tuples in the neighborhood, this is an interesting aspect to explore further. On the other hand, direct algorithms such as DT have the same (worst-case) running time on every input.

Acknowledgements

We acknowledge the support from grant 252-000-303-112.

References

1. Freuder, E.C.: Eliminating interchangeable values in constraint satisfaction problems. In: Proceedings of AAAI 1991, Anaheim, California, pp. 227–233 (1991)
2. Beckwith, A.M., Choueiry, B.Y., Zou, H.: How the level of interchangeability embedded in a finite constraint satisfaction problem affects the performance of search. In: Proceedings of the 14th Australian Joint Conference on AI, pp. 50–61 (2001)

3. Lal, A., Choueiry, B.Y., Freuder, E.C.: Neighborhood interchangeability and dynamic bundling for non-binary finite csps. In: Proceedings of AAAI 2005, Pittsburgh, Pennsylvania, pp. 397–404 (2005)
4. Weigel, R., Faltings, B.V.: Compiling constraint satisfaction problems. Artificial Intelligence 115, 257–287 (1999)
5. Haselböck, A.: Exploiting interchangeabilities in constraint satisfaction problems. In: Proceedings of IJCAI 1993, Chambéry, France, pp. 282–287 (1993)
6. Cheng, K.C.K., Yap, R.H.C.: Maintaining generalized arc consistency on ad hoc r-ary constraints. In: Stuckey, P.J. (ed.) CP 2008. LNCS, vol. 5202. Springer, Heidelberg (2008)
7. Lecoutre, C., Szymanek, R.: Generalized arc consistency for positive table constraints. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 284–298. Springer, Heidelberg (2006)
8. Choueiry, B.Y., Davis, A.M.: Dynamic bundling: Less effort for more solutions. In: Koenig, S., Holte, R.C. (eds.) SARA 2002. LNCS (LNAI), vol. 2371, pp. 64–82. Springer, Heidelberg (2002)
9. Bellicha, A., Capelle, C., Habib, M., Kokeny, T., Vilarem, M.C.: Csp techniques using partial orders on domains values. In: ECAI 2004 Workshop on constraint satisfaction issues raised by practical applications, Amsterdam, The Netherlands, pp. 47–56 (1994)

Infeasibility Driven Evolutionary Algorithm (IDEA) for Engineering Design Optimization

Hemant K. Singh, Amitay Isaacs, Tapabrata Ray, and Warren Smith

School of Aerospace, Civil and Mechanical Engineering,
University of New South Wales, Australian Defence Force Academy, Canberra, ACT
{h.singh,a.isaacs,t.ray,w.smith}@adfa.edu.au
http://www.unsw.adfa.edu.au

Abstract. Engineering design often requires solutions to constrained optimization problems with highly nonlinear objective and constraint functions. The optimal solutions of most design problems lie on the constraint boundary. In this paper, Infeasibility Driven Evolutionary Algorithm (IDEA) is presented that searches for optimum solutions near the constraint boundary. IDEA explicitly maintains and evolves a small proportion of infeasible solutions. This behavior is fundamentally different from the current state of the art evolutionary algorithms, which rank the feasible solutions higher than the infeasible solutions and in the process approach the constraint boundary from the feasible side of the design space. In IDEA, the original constrained minimization problem with k objectives is reformulated as an unconstrained minimization problem with $k+1$ objectives, where the additional objective is calculated based on the relative amount of constraint violation among the population members. The presence of infeasible solutions in IDEA leads to an improved rate of convergence as the solutions approach the constraint boundary from both feasible and infeasible regions of the search space. As an added benefit, IDEA provides a set of marginally infeasible solutions for trade-off studies. The performance of IDEA is compared with Non-dominated Sorting Genetic Algorithm II (NSGA-II) [1] on a set of single and multi-objective mathematical and engineering optimization problems to highlight the benefits.

1 Introduction

Most real life engineering design problems are constrained optimization problems. The solutions to these problems often lie along the constraint boundary. It is also known that the performance of the evolutionary algorithms for constrained optimization is largely dependent on the mechanism for constraint handling.

A detailed review of various constraint handling techniques used with evolutionary algorithms is presented in [2]. Penalty function based methods and their variants are the most commonly adopted form of constraint handling, where the fitness of an infeasible solution is degraded by a weighted sum of individual constraint violations. However, penalty function based methods are known to be highly sensitive to the choice of penalty parameters. Dynamic setting of penalty parameters has been studied in literature, and most of the implementations rely on certain predefined parameter update rules [3,4] that may not work for all problems. Runarsson and Yao [5] introduced a stochastic ranking procedure to strike a balance between objective and penalty functions.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 104–115, 2008.

Other conventional approaches for constraint handling include the use of repair algorithms [6,7]. The drawback of repair mechanisms is that they have to be designed specifically for each problem. Incorporation of heuristic rules such as linear ranking [8] and binary tournament [9] have been applied successfully, but a complementary mechanism has to be used to preserve diversity.

Fundamentally, most evolutionary algorithms such as NSGA-II, prefer a feasible solution over an infeasible solution. However, for many engineering design problems, an infeasible solution near the constraint boundary is desirable over a feasible solution away from it. This idea was incorporated in the method proposed by Vieira et al. [10], who treated constraints as objectives during the search. A similar attempt to maintain infeasible solutions for trade-off studies from a system design perspective also appears in the works of Saxena and Deb [11]. Ray, Tai and Seow [12] used the non-dominated ranks with respect to objectives and constraints (separately and together) to determine the fitness of the solutions for constrained problems. The main drawback of treating constraints as objectives is the considerable computational cost for non-dominated sorting with large number of constraints. A comparative study on using multi-objective techniques to solve constraint optimization problems can be found in [13].

In recent studies, *two-market* algorithms have been suggested, where each generation consists of two phases: *optimality improvement*, which is aimed to improve objective function values, followed by *feasibility improvement*, which reduces constrained violations [14]. Hingston et al. [15] proposed *objective first* ranking scheme and compared it with widely used *feasibility first* scheme. They claimed that their scheme worked better than the latter as it maintained infeasible solutions for longer duration during the search, thereby exploring the search space more effectively.

Clearly, the recent studies emphasize that it is advantageous to maintain infeasible solutions during the search for more efficient exploration of the search space. Consequently, extraction of information from infeasible points for optimization has attracted significant attention of researchers in evolutionary computation. Isaacs, Ray and Smith [16] introduced a Constraint Handling Evolutionary Algorithm (CHEA) that obtains the constrained as well as the unconstrained optima simultaneously. The incorporation of a search through the infeasible space resulted in an improved rate of convergence. In this paper, CHEA is extended as Infeasibility Driven Evolutionary Algorithm (IDEA) to deliver (a) a set of non-dominated solutions close to the Pareto optimal solutions for multi-objective problems (b) a few *marginally* infeasible solutions for trade-off studies, and (c) an improvement in the rate of convergence that is of great importance computationally expensive design problems. The performance of IDEA is compared with NSGA-II on a number of single objective and multi-objective test problems and two engineering design problems (bulk carrier ship design and car side impact problem).

The rest of the paper is organized as follows. The proposed algorithm IDEA is described in Sect. 2. Provided in Sect. 3 are the results of the numerical test problems and engineering test problems. The trade-off study for test problem g06 is illustrated in Sect. 4. In Sect. 5, parametric studies are given, followed by a summary and conclusion in Sect. 6.

2 Infeasibility Driven Evolutionary Algorithm (IDEA)

A multi-objective optimization problem can be formulated as shown in (1).

$$\begin{aligned} \text{Minimize} \quad & f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}) \\ \text{Subject to} \quad & g_i(\mathbf{x}) \geq 0, \quad i = 1, \ldots, m \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ is the design variable vector bounded by lower and upper bounds $\mathbf{x} \in \mathbf{S} \subset \mathbb{R}^n$. A single objective optimization problem follows the same formulation with $k = 1$.

To effectively search the search space (including the feasible and the infeasible regions), the original k objective constrained optimization problem is reformulated as $k+1$ objective unconstrained optimization problem as given in (2).

$$\begin{aligned} \text{Minimize} \quad & f_1'(\mathbf{x}) = f_1(\mathbf{x}), \ldots, f_k'(\mathbf{x}) = f_k(\mathbf{x}) \\ & f_{k+1}'(\mathbf{x}) = \text{Violation measure} \end{aligned} \tag{2}$$

The additional objective represents a measure of constraint violation, which is referred to as "violation measure" in this study. It is based on the amount of relative constraint violation among the population members. Each solution in the population is assigned m ranks, corresponding to m constraints. The ranks are calculated as follows. To get the ranks corresponding to ith constraint, all the solutions are sorted based on the constraint violation value of ith constraint. Solutions that do not violate the constraint are assigned rank 0. The solution with the least constraint violation value gets rank 1, and the rest of the solutions are assigned increasing ranks in the ascending order of the constraint violation value. The process is repeated for all the constraints and as a result each solution in the population gets assigned m ranks. The violation measure is the sum of these m ranks corresponding to m constraints. The process of assignment of ranks and calculation of violation measure is illustrated in Table 1.

Table 1. Calculation of constraint violation measure for sample population with 10 solutions

	Violation Value			Relative ranks			Violation
Solution	C1	C2	C3	C1	C2	C3	Measure
1	3.50	90.60	8.09	3	7	7	17
2	5.76	7.80	6.70	4	5	5	14
3	–	3.40	7.10	0	3	6	9
4	1.25	–	0.69	1	0	1	2
5	13.75	90.10	5.87	6	6	4	16
6	100.70	2.34	3.20	7	2	2	11
7	–	5.09	4.76	0	4	3	7
8	1.90	–	–	2	0	0	2
9	–	110.56	–	0	8	0	8
10	8.90	2.30	9.80	5	1	8	14

The main steps of IDEA are outlined in Algorithm 1. IDEA uses simulated binary crossover (SBX) and polynomial mutation operators [1] to generate offspring from a pair of parents selected using binary tournament as in NSGA-II. Individual solutions in the population are evaluated using the original problem definition (1) and infeasible solutions are identified. The solutions in the parent and offspring population are divided into a feasible set (S_f) and an infeasible set (S_{inf}). The solutions in the feasible set and the infeasible set are both ranked separately using the non-dominated sorting and crowding distance sorting [1] based on $k+1$ objectives as per the modified problem definition (2). The solutions for the next generation are selected from both the sets to maintain infeasible solutions in the population. In addition, the infeasible solutions are ranked higher than the feasible solutions (described below) to provide a selection pressure to create *better* infeasible solutions resulting in an active search through the infeasible search space.

Algorithm 1. Infeasibility Driven Evolutionary Algorithm (IDEA)

Require: N {Population Size}
Require: $N_G > 1$ {Number of Generations}
Require: $0 < \alpha < 1$ {Proportion of infeasible solutions}
1: $N_{inf} = \alpha * N$
2: $N_f = N - N_{inf}$
3: pop_1 = Initialize()
4: Evaluate(pop_1)
5: **for** $i = 2$ to N_G **do**
6: $childpop_{i-1}$ = Evolve(pop_{i-1})
7: Evaluate($childpop_{i-1}$)
8: $(S_f, S_{inf}) =$ Split($pop_{i-1} + childpop_{i-1}$)
9: Rank(S_f)
10: Rank(S_{inf})
11: $pop_i = S_{inf}(1 : N_{inf}) + S_f(1 : N_f)$
12: **end for**

A user-defined parameter α is used to maintain a set of infeasible solutions as a fraction of the size of the population. The numbers N_f and N_{inf} denote the number of feasible and infeasible solutions as determined by parameter α. If the infeasible set S_{inf} has more than N_{inf} solutions, then first N_{inf} solutions are selected based on their rank, else all the solutions from S_{inf} are selected. The rest of the solutions are selected from the feasible set S_f, provided there are at least N_f number of feasible solutions. If S_f has fewer solutions, all the feasible solutions are selected and the rest are filled with infeasible solutions from S_{inf}. The solutions are ranked from 1 to N in the order they are selected. Hence, the infeasible solutions selected first will be ranked higher than the feasible solutions selected later. A fixed value of $\alpha = 0.2$ has been used in the experiments presented on benchmark problems. Later in the paper, variation in the performance of IDEA with change in the value of α has been discussed.

3 Numerical Examples

The performance of proposed algorithm IDEA is studied on a number of benchmark test functions (g-series and CTP) and two engineering design problems. The results and comparison with NSGA-II are given in the following subsections.

3.1 G-Series Problems

For studying performance on single objective optimization problems, g-series test function suite [17] has been chosen. Only the problems without equality constraints have been considered. A population of size 200 is allowed to evolve over 1750 generations. To observe the behavior across multiple runs and multiple parameters settings, independent runs are performed with two values for following parameters: probability of crossover (0.8, 0.9), probability of mutation (0.1, 0.2), distribution index of crossover (15, 20), distribution index of mutation (20, 30). For each parameter combination, two runs are done with different random seeds, thus resulting in a total of $2^5 = 32$ runs.

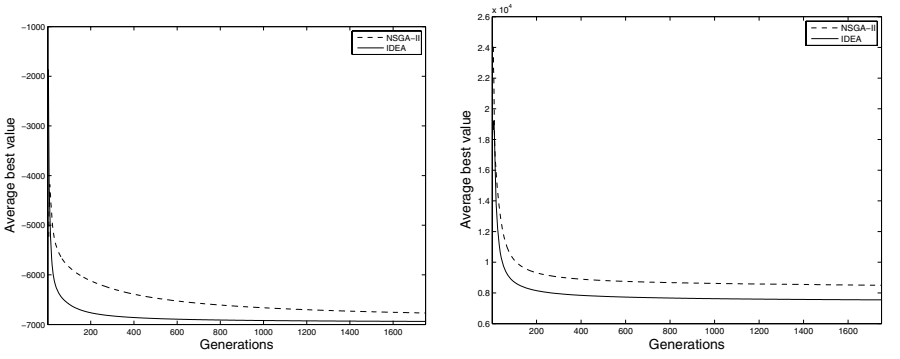
For problems g01, g02, g04, g06 – g10, and g12, the best objective values averaged over 32 runs are listed in Table 2. For a fixed number of function evaluations, IDEA consistently reports a better objective value as compared to NSGA-II. Shown in Fig. 1 are the progress plots for problems g06 and g10 depicting the best solution averaged over all the runs across generations. The other problems show a similar trend and are omitted for sake of brevity. It is clear that the average convergence rate and final results using IDEA are better than NSGA-II.

Table 2. The results of g-series problems averaged over 32 runs using NSGA-II and IDEA

	NSGA-II Results				IDEA Results			
	Best	Mean	Worst	S.D.	Best	Mean	Worst	S.D.
g01	-15.0000	-14.5979	-12.4457	0.8755	-15.0000	**-14.9997**	-14.9988	0.0003
g02	0.8033	0.7943	0.7640	0.0081	0.8032	**0.8019**	0.7934	0.0022
g04	-30665.30	-30661.21	-30618.70	8.4164	-30665.50	**-30665.47**	-30665.30	0.0523
g06	-6946.550	-6921.696	-6892.390	14.0538	-6961.790	**-6961.473**	-6960.690	0.265
g07	24.4532	25.8522	31.9884	1.7400	24.3811	**25.0916**	27.1796	0.6293
g08	-0.09582	-0.09582	-0.09582	0	-0.09582	-0.09582	-0.09582	0
g09	680.6450	681.1611	682.2540	0.4398	680.6670	**680.9331**	681.4170	0.1967
g10	7355.190	8284.448	10030.100	0.0118	7113.430	**7434.930**	7778.320	0.0000
g12	1.000	1.000	1.000	0	1.000	1.000	1.000	0

3.2 CTP Problems

CTP problems [18] are a set of constrained bi-objective optimization problems. Fifteen independent runs are performed on CTP2 – CTP8 using NSGA-II and IDEA by varying the random seed. Following parameters are kept fixed: probability of crossover is 0.9, probability of mutation is 0.1, distribution index for crossover is 15 and distribution



(a) g06 **(b)** g10

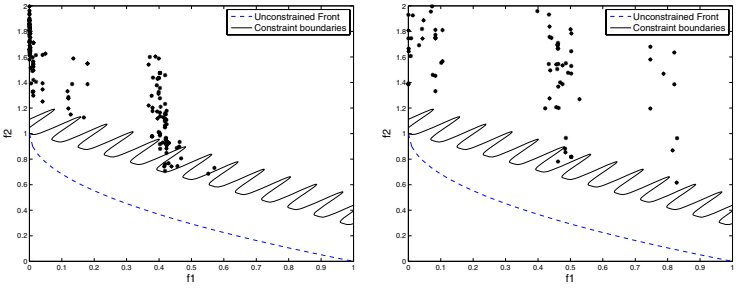
Fig. 1. Progress of the average of the best objective values (over 32 runs) for problems g06 and g10 obtained by NSGA-II and IDEA

index for mutation is 20. A population of size 200 is evolved over 200 generations. Two performance metrics, *Displacement* [19] and *Hypervolume* [20] are used to compare the results obtained by both algorithms. The reference point used for calculating *Hypervolume* for CTP6 and CTP8 was (2,20), while for the rest of the CTP problems it was (2,2).

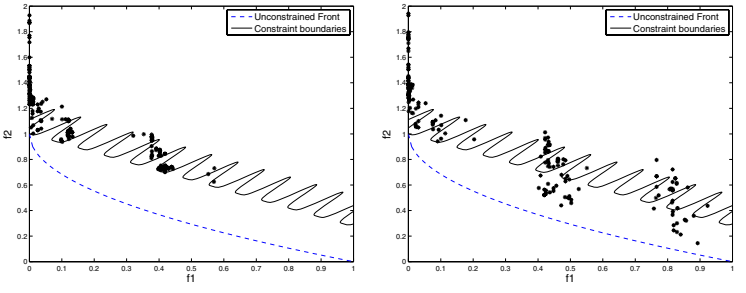
The performance metrics averaged over all the runs using NSGA-II and IDEA for CTP problems are listed in Table 3 and Table 4. It is seen that the *average Displacement* metric values obtained by IDEA are significantly lower than NSGA-II for all CTP problems (Table 3). Similarly, the *average Hypervolume* values obtained by IDEA are consistently higher than NSGA-II. The poor performance of NSGA-II is due to the tendency to converge to sub-optimal fronts for CTP problems. The evolution of population for the problem CTP2 using NSGA-II and IDEA is shown in Fig. 2. For NSGA-II, the population approaches the Pareto optimal solutions from the feasible search space and has difficulty in searching close to the constraint boundary. IDEA, on the other hand, maintains the population in both the feasible and the infeasible space, thus capturing the entire Pareto optimal set much faster.

3.3 Car Side Impact Problem

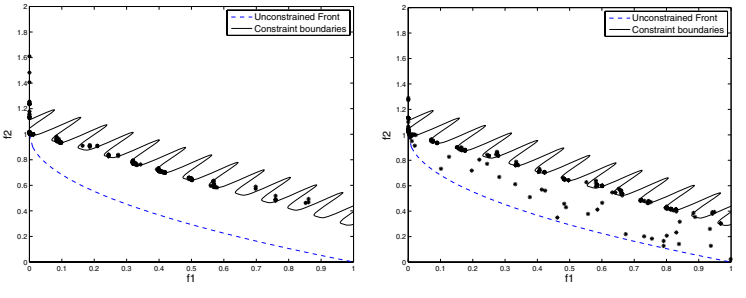
Car side impact problem is a single objective problem, where the objective is to minimize the weight of the car subject to constraints on safety performance characteristics when subjected to side impact. The problem formulation can be found in [11]. Thirty-two runs of NSGA-II and IDEA are performed for the problem using same parameters as used for g-series test functions. Population of size 100 is evolved for 200 generations for both algorithms. The summary of the results obtained by NSGA-II and IDEA are shown in Table 5. The average performance of IDEA is better than NSGA-II as seen from the lower mean and standard deviation of the objective value (Table 5). Also, a faster convergence rate is observed from Fig. 3(a).



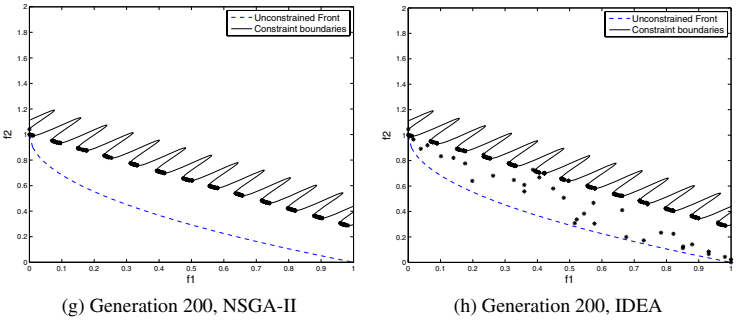
(a) Generation 25, NSGA-II (b) Generation 25, IDEA



(c) Generation 30, NSGA-II (d) Generation 30, IDEA



(e) Generation 50, NSGA-II (f) Generation 50, IDEA



(g) Generation 200, NSGA-II (h) Generation 200, IDEA

Fig. 2. Evolution of NSGA-II and IDEA population over generations for CTP2 (Population size is 200)

Table 3. Averaged *Displacement* metric for CTP results obtained using NSGA-II and IDEA

	NSGA-II Results		IDEA Results	
	Mean	S.D.	Mean	S.D.
CTP2	0.010207	0.010288	**0.001255**	0.004361
CTP3	0.029705	0.037654	**0.007173**	0.017421
CTP4	0.074832	0.048535	**0.027846**	0.016881
CTP5	0.005649	0.005705	**0.001363**	0.004037
CTP6	0.022218	0.071536	**0.000871**	0.001302
CTP7	0.010337	0.014110	**0.008061**	0.017948
CTP8	0.134973	0.150119	**0.004300**	0.010169

Table 4. Averaged *Hypervolume* metric for CTP results obtained using NSGA-II and IDEA

	NSGA-II Results		IDEA Results	
	Mean	S.D.	Mean	S.D.
CTP2	2.752907	0.303290	**3.011504**	0.180320
CTP3	2.776672	0.279543	**2.957526**	0.166232
CTP4	2.314839	0.362156	**2.764931**	0.137807
CTP5	2.630965	0.311307	**2.950680**	0.164526
CTP6	35.765456	3.076219	**36.768927**	0.104876
CTP7	3.147462	0.641003	**3.254253**	0.813259
CTP8	30.880969	5.700612	**35.989817**	0.435925

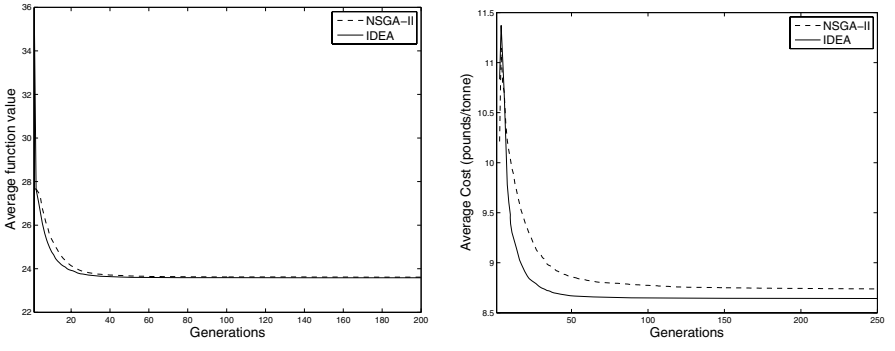
3.4 Bulk Carrier Design Problem

The bulk carrier design problem was originally formulated by Sen and Yang [21]. The same formulation with corrections for Froude number has been presented in [22]. The original problem has three objectives: (1) Minimization of transport cost, (2) Minimization of light ship mass and (3) Maximization of annual cargo transport capacity. Studies on two different formulations of the problem are presented in this paper.

1. A single objective design problem, where only the minimization of transport cost is considered. Along with the original constraints, an additional constraint on the minimum cargo transported (10^6 tonnes/year) has been imposed. The summary of results of 32 independent runs (with the same parameter values as used for g-series functions) are given in Table 5. It is seen that the best and the average objective values obtained by IDEA are better than NSGA-II. A faster convergence rate can be seen in Fig. 3(b).
2. A two objective design problem, where minimization of transport cost and maximization of annual cargo transport capacity are considered. The performance metrics averaged over multiple runs of final solutions obtained using NSGA-II and IDEA are listed in Table 6. For the *Hypervolume* metric, the reference point used is (15, -0.5e6). Lower value of average *Displacement* metric for IDEA implies that

Table 5. Results for car side impact problem and bulk carrier design problem (single objective formulation)

	NSGA-II Results				IDEA Results			
	Best	Mean	Worst	S.D.	Best	Mean	Worst	S.D.
side impact	23.5857	23.6146	24.0059	0.1008	23.5857	**23.5866**	23.5942	0.0018
bulk carrier	8.6206	8.7382	8.8992	0.0585	8.6083	**8.6424**	8.7568	0.038680



(a) Car side impact **(b)** Bulk carrier design

Fig. 3. Average of best feasible function values over 32 runs at each generation, obtained using NSGA-II and IDEA

Table 6. Performance metrics for Bulk carrier design problem (two objective formulation)

	NSGA-II Results		IDEA Results	
	Mean	S.D.	Mean	S.D.
Displacement	1829.044	2356.498	**530.115**	740.564
Hypervolume	8218824.76	328529.05	**8418500.64**	99847.06

the non-dominated front obtained by IDEA is closer to the Pareto optimal set as compared to that obtained using NSGA-II. Similarly, higher average value of *Hypervolume* metric indicates the non-dominated solutions are distributed closer to the Pareto optimal set as in the case of IDEA.

4 Trade-Off Studies

In engineering design, one is often interested to know if a significant improvement in objective function can be achieved by relaxing one (or more) of the constraints marginally. Since IDEA maintains solutions close to constraint boundaries during the search, The final population obtained after the run contains solutions that can provide

Table 7. Marginally infeasible solutions obtained using IDEA for g06

x_1	x_2	$f(\mathbf{x})$	Violations C1	C2
14.095100	0.840314	-6964.723300	0	0.023632
14.095100	0.841393	-6963.535085	0	0.014656
14.062200	0.772189	-7041.657047	0.002145	0.063455
14.096500	0.792284	-7017.680063	0	0.448186
14.095100	0.803637	-7005.192343	0	0.330106

useful trade-off information. Few of the infeasible solutions in the final population obtained using IDEA for the problem g06 are listed in Table 7. The minimum objective value for g06 is -6961.83188. The objective can be improved substantially by relaxing one or both the constraints marginally, as seen from the Table 7.

5 Variation in Performance with Infeasibility Ratio

To see the effect of the infeasibility ratio (α) on the performance of IDEA, fifteen independent runs each were made with different values of α for the problem g06. The average convergence plot is shown in Fig. 4. Parameters used were: population size = 200, number of generations = 1750, probability of crossover = 0.9, probability of mutation = 0.1, distribution index of crossover = 15, distribution index of mutation = 20. It is seen that the performance of IDEA is consistent over a wide range of α. Even by maintaining small proportion ($\alpha = 0.05$) of infeasible solutions in the population, significant improvement can be achieved in the convergence rate. For $\alpha = 0$, the performance of

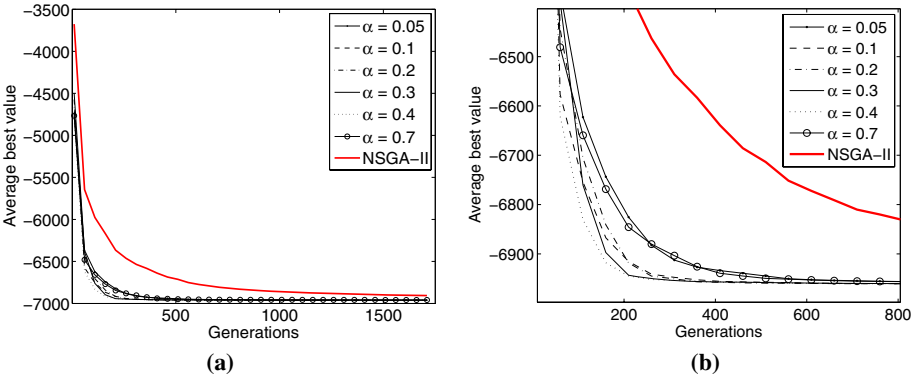


Fig. 4. Variation of IDEA performance for problem g06 with change in α: (a) Over all the generations, (b) During initial generations

IDEA be the same as that of NSGA-II. For multi-objective problems, a high value of α would lead to fewer solutions covering the Pareto-front, and hence is not recommended.

6 Summary and Conclusions

A Infeasibility Driven Evolutionary Algorithm (IDEA) for constrained engineering design optimization problems is presented. The previous work by Isaacs, Ray and Smith on Constraint Handling Evolutionary Algorithm (CHEA) [16] is extended by incorporating a better constraint handling approach. CHEA focused on obtaining the solutions to the original constrained as well as the unconstrained formulation (by dropping the constraints) of the problem. IDEA delivers solutions to the constrained optimization problem and marginally infeasible solutions that are much better suited for the trade-off studies.

The performance of IDEA is compared with NSGA-II on a set of single and multi-objective test problems and two engineering design optimization problems. The results clearly indicate that IDEA has an improved rate of convergence over NSGA-II and the performance is consistent across all problems studied in this paper. Thus, IDEA is a well suited algorithm for constrained design optimization problems.

Acknowledgments

The presented work was supported by grants from Defence and Security Applications Research Center (DSARC), UNSW@ADFA, Canberra, Australia.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation. IEEE Transactions 6, 182–197 (2002)
2. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
3. Joines, J., Houck, C.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. Evolutionary Computation. IEEE World Congress on Computational Intelligence. In: Proceedings of the First IEEE Conference, June 1994, vol. 2, pp. 579–584 (1994)
4. Kazarlis, S., Petridis, V.: Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 211–220. Springer, Heidelberg (1998)
5. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Compution 4, 284–294 (2000)
6. Xiao, J., Michalewicz, Z., Trojanowski, K.: Adaptive Evolutionary Planner/Navigator for Mobile Robots. IEEE Transactions on Evolutionary Computation 1(1), 18–28 (1997)
7. Michalewicz, Z., Xiao, J.: Evaluation of Paths in Evolutionary Planner/Navigator. In: Proceedings of the 1995 International Workshop on Biologically Inspired Evolutionary Systems, Tokyo, Japan, May 1995, pp. 45–52 (1995)

8. Powell, D., Skolnick, M.M.: Using genetic algorithms in engineering design optimization with non-linear constraints. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA 1993), San Mateo, California, pp. 424–431. Morgan Kaufmann Publishers, San Francisco (1993)
9. Deb, K.: An Efficient Constraint Handling Method for Genetic Algorithms. Computer Methods in Applied Mechanics and Engineering 186(2), 311–338 (2000)
10. Vieira, D.A.G., Adriano, R.L.S., Krahenbuhl, L., Vasconcelos, J.A.: Handing constraints as objectives in a multiobjective genetic based algorithm. Journal of Microwaves and Optoelectronics 2(6), 50–58 (2002)
11. Saxena, D.K., Deb, K.: Trading on infeasibility by exploiting constraints criticality through multi-objectivization: A system design perspective. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2007), September 25-28, 2007, pp. 919–926 (2007)
12. Ray, T., Tai, K., Seow, K.: Multiobjective design optimization by an evolutionary algorithm. Engineering Optimization 33(4), 399–424 (2001)
13. Mezura-Montes, E., Coello Coello, C.A.: Constrained Optimization via Multiobjective Evolutionary Algorithms. In: Knowles, J., Corne, D., Deb, K. (eds.) Multiobjective Problems Solving from Nature: From Concepts to Applications. Natural Computing Series. Springer, Heidelberg (2008)
14. Kimbrough, S.O., Lu, M., Wood, D.H., Wu, D.-J.: Exploring a Two-Population Genetic Algorithm. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 1148–1159. Springer, Heidelberg (2003)
15. Hingston, P., Barone, L., Huband, S., While, L.: Multi-level Ranking for Constrained Multiobjective Evolutionary Optimisation. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 563–572. Springer, Heidelberg (2006)
16. Isaacs, A., Ray, T., Smith, W.: Blessings of maintaining infeasible solutions for constrained multi-objective optimization problems. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong, pp. 2785–2792 (2008)
17. Koziel, S., Michalewicz, Z.: Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. Evolutionary Computation 7(1), 19–44 (1999)
18. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley and Sons Pvt. Ltd., Chichester (2001)
19. Bandyopadhyay, S., Saha, S., Maulik, U., Deb, K.: A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE Transactions on Evolutionary Computation (2007)
20. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
21. Sen, P., Yang, J.: Multiple criteria decision support in engineering design. Springer, London (1998)
22. Parsons, M., Scott, R.: Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. Journal of Ship Research 48(1), 61–76 (2004)

Constraint-Based Multi-agent Path Planning

Malcolm Ryan

Centre for Autonomous Systems
School of Computer Science and Engineering
University of New South Wales, Australia

Abstract. Planning collision-free paths for multiple robots traversing a shared space is a problem that grows combinatorially with the number of robots. The naive centralised approach soon becomes intractable for even a moderate number of robots. Decentralised approaches, such as priority planning, are much faster but lack completeness.

Previously I have demonstrated that the search can be significantly reduced by adding a level of abstraction [1]. I first partition the map into subgraphs of particular known structures, such as *cliques*, *halls* and *rings*, and then build abstract plans which describe the transitions of robots between the subgraphs. These plans are constrained by the structural properties of the subgraphs used. When an abstract plan is found, it can easy be resolved into a complete concrete plan without further search.

In this paper, I show how this method of planning can be implemented as a constraint satisfaction problem (CSP). Constraint propagation and intelligent search ordering further reduces the size of the search problem and allows us to solve large problems significantly more quickly, as I demonstrate this in a realistic planning problem based on a map of the Patrick Port Brisbane yard. This implementation also opens up opportunities for the application of a number of other search reduction and optimisation techniques, as I will discuss.

1 Introduction

A major aspect of solving any problem in artificial intelligence (AI) is *knowledge engineering*, that is taking the available background knowledge about a problem and expressing it in a way that it can be exploited by an AI algorithm. This task is crucial to solving any realistically large problem, including the one I address in this paper: multi-agent path planning.

Planning for a single robot, once issues of geometry and localisation have been addressed, becomes a simple matter of finding a path through the *road-map* – the graph G representing the connectivity of free space – between its starting and goal locations. When planning for multiple robots, however, we also need to take into account the possibility for collisions en route. A decentralised approach in which each robot simply planned its own path without reference to the others would not work.

A logical solution is to treat the entire collection of robots as a single entity and use a centralised planner to co-ordinate them. If we again ignore issues of geometry, this equates to finding a path through the *composite graph* $G^k = G \times G \times \ldots \times G$,

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 116–127, 2008.

where k is the number of robots. Each vertex in this graph is a k-tuple of vertices of G representing the positions of each robot. Each edge represents the movement of one robot between neighbouring vertices. Vertices which represent collisions are excluded. A plan is now a path between the vertex representing the robots' initial locations to the vertex representing their goals.

It is easy to see that the size of this graph grows combinatorially with the number of robots. Any algorithm which performs a naive search of the graph will soon require far too much time and memory to complete. A common solution is *prioritised planning* which gives each robot a priority and plan for them in order, with lower priority robots integrating their plans with those of higher priority. This effectively prunes the search space by eliminating certain possibilities (in which higher priority robots go out of their way to allow lower priority robots to pass). Searching this reduced space is much faster, but the pruning may eliminate the only viable solutions, making the algorithm incomplete.

In order to efficiently handle large numbers of robots without sacrificing completeness we need some way to incorporate more knowledge about the domain. In my previous work [1] I have shown how structural information about the road-map can be exploited to significantly reduce search. The map is decomposed into subgraphs of particular known structure, *cliques*, *halls* and *rings*, which place constraints on which robots can enter or leave at a particular time. Planning is done at a level of abstraction, in terms of the configuration of each subgraph and the robots' transitions between them. Once an abstract plan has been constructed the concrete details of robots' movement within each subgraph can be resolved algorithmically, without the need for further search. This approach is proven to be sound and complete.

In this work we extend these previous results by showing how the subgraph planning process can be encoded as a constraint satisfaction problem (CSP). With this formulation, a CSP-solver can make more efficient use of the domain knowledge to prune the search space to a much greater degree allowing us to solve problems significantly larger than before. It also opens up the possibility for optimisation of plans and more complex planning tasks than simple goal achievement.

In the next section I will describe the subgraph planning approach in greater detail. This will be followed by a brief introduction to constraint programming leading into the constraint representation of our planning problem. The efficiency of this new approach will be evaluated on tasks using a map of the Patrick Port Brisbane facility and we will conclude with discussion of related work and future directions.

2 Subgraph Planning

We can formalise our problem as follows. The road-map is provided in the form of a graph $G = (V, E)$ representing the connectivity of free space for a single robot moving around the world (e.g. a vertical cell decomposition or a visibility graph, [2]). We also have a set of robots $R = \{r_1, \ldots, r_k\}$ which we shall consider to be homogeneous, so a single map suffices for them all. All starting locations and goals lie on this road-map.

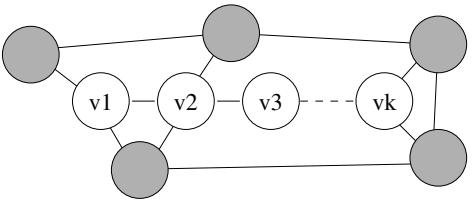


Fig. 1. A hall subgraph

We shall assume that the map is constructed so that collisions only occur when one robot is entering a vertex v at the same time as another robot is occupying, entering or leaving this vertex. Robots occupying other vertices in the map or moving on other edges do not affect this movement. With appropriate levels of underlying control these assumptions can be satisfied for most real-world problems.

The road-map is partitioned into a collection of induced subgraphs $\mathcal{P} = \{S_1, \ldots, S_m\}$ of known structure. In this paper we shall consider only one kind of subgraph: the *hall*. A hall is a singly-linked chain of vertices with any number of entrances and exits, as illustrated in Figure 1. They are commonly found in maps as narrow corridors or roads which may contain several robots but which prevent overtaking. Formally this is represented as $H = \langle v_1, \ldots, v_m \rangle$ with: $(v_i, v_j) \in E$ iff $|i - j| = 1$.

The configuration of a hall can abstract the exact positions of the robots and merely record their order, which cannot be changed without a robot entering or leaving. When a robot enters the hall, a number of different configurations are possible, depending on which edge it uses (at an end or in the middle) and the number of other occupants. When a robot leaves the hall it can simply be removed from the configuration. In either case the order of the other robots in the hall remains the same.

Resolving a step of the abstract plan means shuffling the robots in the hall left or right to either move the departing robot to its exit or to open a space at the appropriate vertex (and position in the sequence of occupants) and for an incoming robot to enter.

An abstract plan is thus an alternating sequence of hall configurations and subgraph transitions. In previous work I have restricted this to a single robot transitioning on each step. The constraint formulation I shall present in this paper allows us to relax this restriction.

3 Constraint Programming

Constraint programming is a methodology for representing and solving combinatorial search problems through constraint propagation and intelligent search. Problems are represented as collections of *variables* over finite domains (usually subsets of the integers) and *constraints* which are relations between the variables that they are required to satisfy. Constraint solvers are designed to represent a large number of different constraints and use them to propagate information from one variable to another so that their domains are consistent (with some degree of strength) with the constraints between them.

Combining constraint propagation with search, we are able to prune the search space of a problem by alternately assigning values to variables and propagating the change to restrict the domains of other unassigned variables. Informed choice of the search order can maximise the benefits of propagation and further reduce the search. For this project I used Gecode/J – a Java-based constraint solver [3].

4 The Constraint Representation

To convert the planning task into a constraint satisfaction problem we need to describe it as a finite set of integer variables. As it stands the task is open ended: a plan can be of any length. To make it finite we need to restrict the plan to a fixed length. If a plan of a given length cannot be found, then a new CSP representing a longer plan can be constructed and the process repeated.[1]

To begin our representation we number each vertex, each robot and each subgraph. Let $V = \{1, \ldots, n\}$ represent the vertices, $R = \{1, \ldots, k\}$ represent the robots and $S = \{1, \ldots, m\}$ represent the subgraphs. Let V_i be the set of vertices for subgraph i. It is useful, as we will see later, to number the vertices so that each V_i contains consecutive integers. Let $\mathcal{E} = \{(a, b) \mid \exists v_a \in V_a, v_b \in V_b, (v_a, v_b) \in E\}$ be the relation defining adjacency between subgraphs. Let L be the length of the abstract plan.

4.1 Abstract Plan Steps

We can now define the variables we need. For each robot $r \in R$ and each step of the plan $i \in \{1 \ldots L\}$ we represent the subgraph it occupies at each time step:

$A_i[r] \in S$ is the index of the subgraph occupied by r at step i,

It is not enough to simply plan in terms of subgraphs. A time step in the abstract plan actually represents multiple concrete actions as the robots are rearranged with each subgraph. We do not need (or wish) to represent all of these concrete steps, but we do need to know where the robots are an the beginning and each of each step, to know what transitions are possible, and what new configurations will result. For this reason, we also create variables representing the first and last vertices in the concrete sub-plan for each robot at each step:

$F_i[r] \in V$ is the index of the first vertex occupied by r at step i,
$T_i[r] \in V$ is the index of the last vertex occupied by r at step i.

We constrain these variables as follows:

Robots can only move between neighbouring subgraphs

$$A_i[r] \neq A_{i+1}[r] \rightarrow (A_i[r], A_{i+1}[r]) \in \mathcal{E} \tag{1}$$

[1] Note that this makes the problem only semi-decideable. There is no sure way to know when no possible plan of any length exists. In practice, this is rarely a problem. Planning stops when plans get beyond a certain maximum length.

$F_i[r]$ and $T_i[r]$ must belong to the given subgraph

$$A_i[r] = a \rightarrow F_i[r] \in V_a \tag{2}$$
$$A_i[r] = a \rightarrow T_i[r] \in V_a \tag{3}$$

Two robots cannot be in the same vertex at the same time

$$distinct(F_i[1], \ldots, F_i[k]) \tag{4}$$
$$distinct(T_i[1], \ldots, T_i[k]) \tag{5}$$

Consecutive sub-plans are linked by valid transitions

$$(T_i[r], F_{i+1}[r]) \in E \tag{6}$$
$$T_i[r_x] \neq F_{i+1}[r_y], \forall r_x \neq r_y \tag{7}$$

No-ops only occur at the end of the plan

$$(\forall r \in R : A_{i-1}[r] = A_i[r]) \rightarrow (\forall r \in R : A_i[r] = A_{i+1}[r]) \tag{8}$$

If a subgraph is full, its occupants cannot move

$$A_i[r] = a \ \wedge \ count_{\rho \in R}(A_i[\rho] = a) = |V_a| \rightarrow F_i[r] = T_i[r] \tag{9}$$

These constraints apply to any abstract plan, regardless of the structure of its subgraphs, but they fail to completely specify the problem. In particular, they do not guarantee that the configuration given by $(T_i[1], \ldots, T_i[k])$ is reachable from $(F_i[1], \ldots, F_i[k])$. To ensure this we must refer to the particular properties of the subgraphs.

4.2 Hall Ordering

In the case of the hall subgraph, we require that the order of robots in the hall does not change between transitions. If r_x is on the left of r_y at the beginning of a sub-plan it must also be so at the end (and vice versa). We can represent this more easily if we number the vertices in the hall consecutively from one end to the other. Then for two robots in the hall, we will require $F_i[r_x] < F_i[r_y] \Leftrightarrow T_i[r_x] < T_i[r_y]$.

It will be useful in the search for a plan to be able to explicitly choose an ordering between two robots without assigning them to particular vertices. To this end, we create a new set of variables to represent the ordering of robots in each sub-plan: $Ord_i[r_x, r_y] \in \{-1, 0, 1\}$. Conveniently we can use one set of variables to describe the configuration of all halls simultaneously, since the value is only important if two robots are in the same subgraph at the same time. If r_x and r_y are in different subgraphs, then $Ord_i[r_x, r_y]$ is 0. Otherwise it must be either -1 or 1, indicating the two possible orderings: r_x before r_y or r_y before r_x.

Formally we add the following constraints:

Robots are ordered iff they are both in the same hall

$$A_i[r_x] \in \mathcal{H} \ \wedge \ A_i[r_x] = A_i[r_y] \Leftrightarrow Ord_i[r_x, r_y] \neq 0 \tag{10}$$

Ordering variables affect concrete positions

$$Ord_i[r_x, r_y] = -1 \rightarrow F_i[r_x] < F_i[r_y] \wedge T_i[r_x] < T_i[r_y] \quad (11)$$

$$Ord_i[r_x, r_y] = 1 \rightarrow F_i[r_x] > F_i[r_y] \wedge T_i[r_x] > T_i[r_y] \quad (12)$$

Ordering variables persist across sub-plan transitions

$$A_i[r_x] = A_{i+1}[r_x] \wedge A_i[r_y] = A_{i+1}[r_y] \rightarrow Ord_i[r_x, r_y] = Ord_{i+1}[r_x, r_y] \quad (13)$$

This completes our description. Any abstract plan which satisfies these constraints can be resolved into a correct concrete plan without further search.

5 Search

Constraint propagation alone will not solve this problem; the constraints are not powerful enough to eliminate every wrong solution. We must also perform a search, experimentally assigning values to variables until a complete plan is found that satisfies all the constraints. By enumerating all the variables at the outset, we are able to assign values to them in any order we wish, unlike standard path-planning algorithms which generally operate in forward temporal order only.

Common wisdom in constraint solving is to assign variables so that failures, if they are going to occur, happen early at shallow levels of the tree so that large amounts of backtracking are avoided. The standard heuristic is to assign the most constrained variables first. In this particular problem it makes sense to assign the subgraph variables $A_i[r]$ first, followed by the order variables $Ord_i[r_x, r_y]$ and finally the transition variables $F_i[r]$ and $T_i[r]$, since each is strongly constrained by the one that comes before. In each case we choose the variable with the smallest domain.

When choosing a value for the variable there are two things to consider: 1) choose a value which is most likely to lead to a solution, 2) choose a value which places the least constraint on other variables. When choosing subgraph values for the $A_i[r]$ variables we apply the first principle by choosing the subgraph which is closest to the next assigned subgraph for robot r (based on a precomputed single-robot all-shortest-paths matrix). If there are two such options, then the subgraph with the fewest occupants is selected, according to the second principle.

The heuristic for selecting the ordering value for $Ord_i[r_x, r_y]$ is to consider the concrete values that it immediately affects $F_i[r_x]$, $T_i[r_x]$, $F_i[r_y]$ and $T_i[r_y]$. For each ordering we can easily compute the resulting domain sizes for each of these variables (ignoring the effect of any other constraints). The ordering which leaves the largest number of alternatives is preferred, by the second principle above.

Finally, values for the concrete steps $F_i[r_x]$ and $T_i[r_x]$ are chosen to minimise the distance between the beginning and end of the plan step.

6 Experiment: The Patrick Port

To evaluate this new planning system I have applied it to a realistic planning problem. Figure 2 shows a map of the Patrick yard at Port Brisbane in Queensland, Australia. This

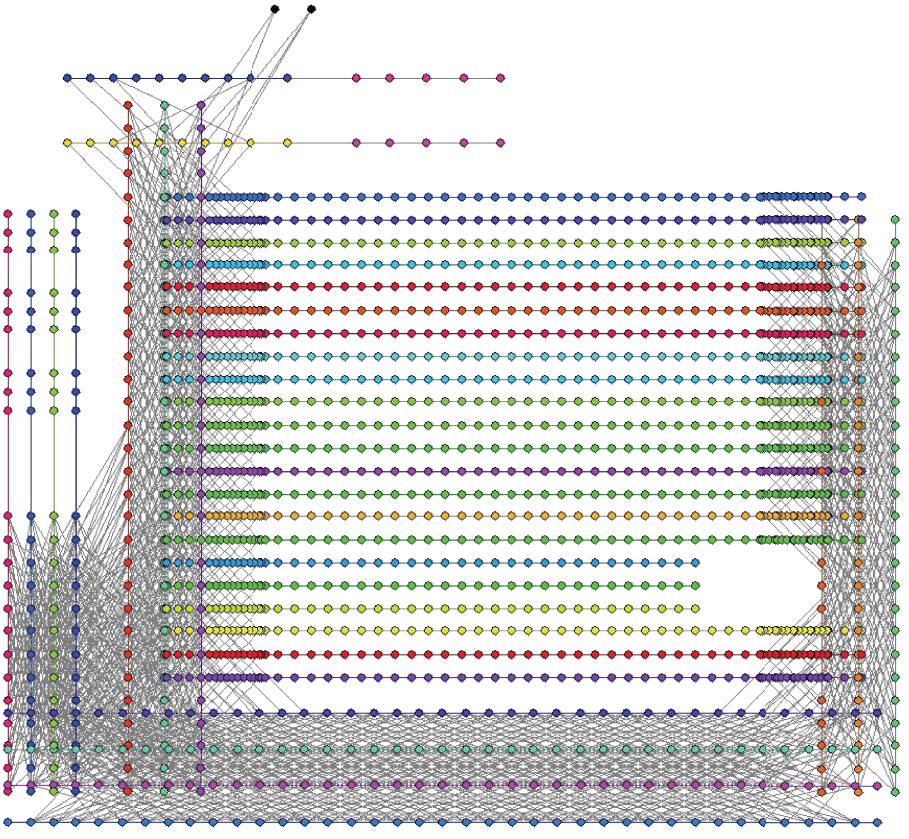


Fig. 2. The map of the Patrick yard at Port Brisbane

map is used to plan the movement of straddle-carriers – enormous, automated vehicles for moving shipping containers around the yard. Efficient, co-ordinated planning for these vehicles is important for the smooth running of the facility.

6.1 The Problem Domain

The map is an undirected graph of 1808 vertices and 3029 edges. The vertices are naturally connected in long straight chains representing the roads around the facility. These roads mean that the vertices can be partitioned into 40 hall subgraphs, with only 2 vertices left over, which must be treated as singletons. The reduced graph has 187 edges connecting neighbouring subgraphs.

This reduced graph was constructed by hand with the aid of a simple interactive tool. Choosing the partition was not difficult; the roads around the port are obvious in the

map and provide a natural set of halls. No effort was made to optimise this partition in any fashion to suit the algorithm.

6.2 Approach

The map was populated with a number of robots which varied from 1 to 40. Each robot was assigned a random initial and final position. A single-robot shortest paths matrix was calculated for the reduced graph and used to calculate the minimum length of the plan as the length of the longest single-robot plan.

A constraint problem was constructed in Gecode/J as described above. The initial and goal states were constrained to their appropriate values and then a search was conducted. An iterative deepening approach was used. A minimum estimate of the plan length was computed by taking maximum shortest path distance for each robot individually. If a plan of this length could not be found, then the length was incremented and the search repeated, until a solution was found or the planner exceeded a 2 minute time limit.

The same problems were also solved with a prioritised planner (also encoded as a CSP in Gecode/J). I compare the results below.

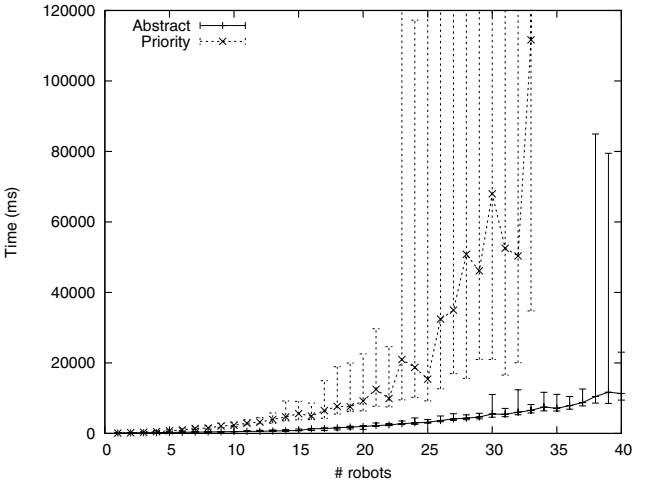
6.3 Results

One hundred different experiments were conducted for each number of robots.[2] The results are plotted in Figures 3(a) and 3(b). The graphs show the median values for total time to construct the CSP and search for a solution and the total memory usage, with whiskers showing the first and third quartiles. Experiments were run with a time limit of 120 seconds and a maximum heap size of 2 gigabytes. Experiments which exceeded these limits are treated as taking infinite time and memory.

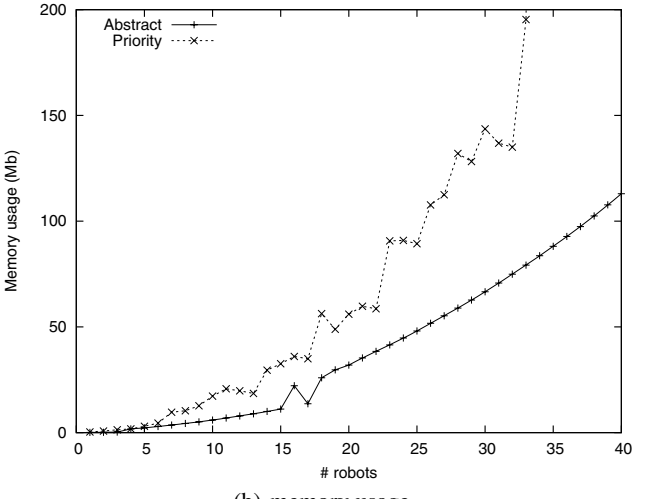
The difference between the two approaches is quite pronounced. The abstract approach shows a much slower rate of increase in both runtime and memory. Performance is comparable for up to 4 robots, but after that point the abstract approach is clearly superior. At 33 robots the graph of the prioritised planner ends because it began to fail more than 50% of the time. The abstract planner was able to handle up to 40 robots, taking only slightly more than 10 seconds in the median case.

There is a noticeable change in both the time and memory graphs around the point of 15 or 16 robots. The explanation for this threshold can be found in the sub-plan sizes. Because of the high connectivity of the graph, two subgraphs randomly chosen can almost always be connected by at most one intermediate subgraph. The probability that more than one is required is small, approximately 4%. As more robots are added to the plan the probability increases that at least one will require a longer plan. The probability reaches 50% about the 15-16 robot mark. So at this point the majority of experiments begin with a plan of 4 abstract steps, while a majority of the smaller problems require only 3. The longer plan requires more variables per robot to represent and thus more time and memory to complete.

[2] Running times were measured on a 3.20GHz Intel(R) Xeon(TM) CPU running Sun JDK 6.0 with 2Gb of heap.



(a) run time



(b) memory usage

Fig. 3. A comparison of median run-times and memory usage for abstract planning and prioritised planning. Error bars show the first and third quartile.

Analysis of median values doesn't give us the full picture. Table 1 shows the number of experiments which failed to complete due to time or memory limits. Many more prioritised experiments failed (23% in total) than abstract experiments (only 1%). In most cases failure occurred because the experiment exceeded the time limit. As stated earlier, constraint-based planning is only semi-decidable, so there is no way to definitely conclude that a problem has no solution, but the data suggests that the incompleteness of the prioritised algorithm may be the issue.

Table 1. Number of failed experiments by problem size. Experiments 1 to 6 showed no failures for either approach.

# Robots	% Failures		# Robots	% Failures		# Robots	% Failures	
	Abs.	Pri.		Abs.	Pri.		Abs.	Pri.
7	0	1	19	0	16	31	4	29
8	0	1	20	2	14	32	4	33
9	0	2	21	0	17	33	4	48
10	0	2	22	0	11	34	2	59
11	0	5	23	0	29	35	1	54
12	0	3	24	1	25	36	7	54
13	0	6	25	0	26	37	3	58
14	0	10	26	2	28	38	9	64
15	0	13	27	0	29	39	7	63
16	0	6	28	0	43	40	6	78
17	1	11	29	2	34			
18	1	19	30	3	46			

7 Conclusion

I have demonstrated how the multi-robot path planning problem can be effectively solved for large numbers of robots by making use of appropriate structural knowledge about the map, in the form of a subgraph decomposition. This knowledge can be encoded precisely as a constraint satisfaction problem and solved using a combination of constraint propagation and heuristic search. This allows us to solve problems of unprecedented size, using time and memory that is significantly smaller than the standard approach of prioritisation.

7.1 Related Work

There has been little previous work in the use of abstractions and modern search techniques in multi-robot path planning. The work that bears most similarity to my own is not explicitly in robot path planning, but in solving the Sokoban puzzle [4,5]. Their division of a map into rooms and tunnels matches to some degree the subgraph decomposition I adopt here. The particular structures they represent are different, but the general ideas of partitioning into independent local subproblems and identifying abstract states from strongly connected components, are the same as those employed in this work. They have not as yet attempted to translate these structures into a formal constraint satisfaction problem.

CSPs have however been applied to a different kind of planning, that is AI task-planning. CPLan [6] directly encodes such planning problems as constraint systems and uses a general purpose constraint solver to find plans. Another approach is to encode the planning graph from an algorithm such as Graphplan [7] and convert it into a CSP, as done in the work of Do and Kambhampati [8] and Lopez and Bacchus [9]. A related approach is the 'planning-as-satisfiability' technique used in planners such as SatPlan [10].

7.2 Future Work

This constraint-based approach opens the door to a number of new possibilities. More complex planning problems can be expressed by adding appropriate constraints to the system. If we extended the representation to include variables for the concrete sub-plans in their entirety, we could add extra constraints to prevent certain vertices from being simultaneously occupied, to add a buffer zone between robots. We could specify goals that involve visiting several locations in sequence. It is already possible to have robots that have no particular goal but to stay out of the way.

If we add variables representing the lengths of the concrete plans, we can begin to work on optimisation. As it stands, the algorithm makes no guarantees that concrete plans will be optimal. Finding perfectly optimal plans is likely to be very time consuming, but a branch-and-bound algorithm could provide a viable alternative, yielding the best plan found in the available time.

This leads us to consider what other advanced CSP-solving techniques could be useful. The most immediately obvious is sub-problem independence [11]. Once the $A_{[}[i]]$[r] variables have been set, the other variables in this problem are partitioned into a number of subsets which do not affect each other. Solving these sub-problems independently could prevent a lot of unnecessary backtracking.

In conclusion, this paper demonstrates the successful combination of domain knowledge and intelligent problem solving tools. It offers not just a fast planning algorithm, but also a validation of constraint programming as an effective knowledge engineering methodology, and one which we should continue to improve upon.

Acknowledgments

I'd like to thank Guido Tack, Mikael Lagerkvist and Christian Schulte from the Gecode development team for all their patient assistance and advice.

References

1. Ryan, M.R.K.: Exploiting subgraph structure in multi-robot path planning. Journal of Artificial Intelligence Research 31, 497–542 (2008)
2. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
3. Gecode Team: Gecode: Generic constraint development environment (2006), `http://www.gecode.org`
4. Botea, A., Müller, M., Schaeffer, J.: Using abstraction for planning in sokoban. In: Schaeffer, J., Müller, M., Björnsson, Y. (eds.) CG 2002. LNCS, vol. 2883, pp. 360–375. Springer, Heidelberg (2003)
5. Junghanns, A., Schaeffer, J.: Sokoban: Enhancing general single-agent search methods using domain knowledge. Artificial Intelligence 129(1-2), 219–251 (2001)
6. van Beek, P., Chen, X.: CPlan: A constraint programming approach to planning. In: Proceedings of the AAAI National Conference, pp. 585–590 (1999)
7. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence 90(1-2), 281–300 (1997)
8. Do, M., Kambhampati, S.: Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. Artificial Intelligence 132(2), 151–182 (2001)

9. Lopez, A., Bacchus, F.: Generalizing GraphPlan by Formulating Planning as a CSP. In: Proceeding of the International Joint Conference on Artificial Intelligence, IJCAI 2003 (2003)
10. Kautz, H., Selman, B., Hoffmann, J.: SatPlan: Planning as Satisfiability. In: Abstracts of the 5th International Planning Competition (2006)
11. Mann, M., Tack, G., Will, S.: Decomposition during search for propagation-based constraint solvers. Technical Report arXiv:0712.2389v2, Cornell University Library (2007)

An Optimality Principle for Concurrent Systems

Langford B. White and Sarah L. Hickmott

School of Electrical and Electronic Engineering
The University of Adelaide
Australia
{lwhite,shick}@eleceng.adelaide.edu.au

Abstract. This paper presents a formulation of an optimality principle for a new class of concurrent decision systems formed by products of deterministic Markov decision processes (MDPs). For a single MDP, the optimality principle reduces to the usual Bellman's equation. The formulation is significant because it provides a basis for the development of optimisation algorithms for decentralised decision systems including a recently proposed method based on Petri Net unfoldings.

1 Introduction

Markov decision processes have been widely studied and applied to a large number of optimisation problems, particularly in planning and scheduling applications (see eg [1], [2]). Many solution techniques have been developed but all basically rely on Bellman's *principle of optimality.* As we shall see, this principle is really quite straightforward and intuitive in nature.

We have a considerable interest in optimisation of concurrent systems. These systems don't have a global clock, so a class of models such as MDPs which are based on a global clock, are not appropriate for concurrent systems. In particular, we are interested in optimisation of automated planning systems which are inherently concurrent in nature. Much of the work which has been done in concurrent planning effectively translates the problem into a (huge) MDP by placing an artificial total ordering on the system. This can lead to excessively large problems which don't scale well.

Recently, a solution to optimisation of concurrent systems based on Petri Nets (PNs) has been proposed [3],[4]. The algorithm, known as ERV-Fly finds globally optimal solutions via a technique known as directed unfolding, described in [5]. A close analysis of the algorithm shown that it embodies (among other things) a principle of optimality for concurrent systems that is not unlike Bellman's principle for MDPs in nature. The purpose of this paper is to derive this principle and show that it is a natural extension of the totally ordered (MDP) case. The result, although appearing simple in hindsight, is significant because it provides a necessary condition that any optimisation algorithm must meet. Also, the principle may lead to other optimisation algorithms for concurrent systems, much in the way that Bellman's principle has for MDPs. We believe that this is the first concise statement of such a principle for a class of concurrent systems.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 128–137, 2008.

The layout of the paper is as follows. We firstly review Bellman's principle for MDPs in a somewhat non-standard fashion, that motivates the generalisation to concurrent systems in a clear way. We then develop a PN model for MDPs. In section 3, we describe how to generalise these PN models to a class of concurrent systems described as *products of MDPs* (PMDPs). Products of transition systems (the Markov part of MDP) have been studied in detail in [6], although not from an decision theoretic point of view. Thus, PMDPs, in some sense, are a new class of models for concurrent decision systems. In section 4, we introduce PN unfoldings as a framework for deriving and proving an optimality principle for, firstly a single MDP, and secondly for a PMDP.

2 Petri Net Models for Markov Decision Processes

A (deterministic) Markov Decision Process (MDP) is a tuple $\mathcal{M} = (S, A, is, g, \pi, \sigma, \phi)$ where

- S is a finite set of *states*,
- A is a finite set of *actions*,
- $is \in S$ is the *initial state*,
- $g \in S$ is the *goal state*,
- $\pi : A \to S$ giving the unique predecessor state to an action,
- $\sigma : A \to S$ giving the unique successor state to an action, and
- $\phi : A \to \mathbb{R}^+$ is the *cost function*.

The semantics of the MDP are as follows. Suppose the system is in some state s, then each action in $\pi^{-1}(s)$ is enabled. Suppose we execute action $a \in \pi^{-1}(s)$, then the system moves to state $s' = \sigma(a)$. We say that $s = \pi(a)$ is the predecessor state for a, and $s' = \sigma(a)$ is the successor state for a. [1] Every time an action a is executed, an additive cost function is incremented by an amount $\phi(a) > 0$. Our objective is to find a sequence of actions $\{a_k\}$ starting at state is and terminating at state g [2] which minimises the total cost. Such a problem is called a *shortest path* problem, and $\{a_k\}$ is called a shortest path. Standard tools (eg value iteration, policy iteration, linear programming) can be applied to this problem. Most solution methods are based around solving *Bellman's equation*, an expression of the *principle of optimality* [1]. A *policy* is a mapping $\psi : S \to A$ which specifies which action a is executed when the system is in state s. The solution of Bellman's equation yields a policy which minimises total cost to reach the goal. One interpretation of Bellman's principle is that on any globally optimal path (sequence of actions) from is to g, each local policy must also be optimal in the sense of choosing an action which minimises the total cost of it

[1] In our model, we only address *deterministic* outcomes. Thus the successor state is uniquely defined by the action being executed. In the general case, there is a probability distribution specified on a set of successor states but we don't consider this case here.

[2] We can easily generalise to the case there the goal consists of a subset of states.

and all past actions. More precisely, if $J^*(s)$ denotes the optimal cost to arrive at a state s from is, then

$$J^*(s) = \min_{a \in \sigma^{-1}(s)} \left(J^*(\pi(a)) + \phi(a) \right) \ , \tag{1}$$

and a minimising action a in (1) specifies the optimal (local) policy to be executed in state $\pi(a)$ to arrive in state s. [3]

Let A^* denote the set of all sequences of actions (called *words*), then for any $v = \{a_i\} \in A^*$ we define the cost of v by extending $\phi : A^* \to \mathbb{R}^+ \cup \{\infty\}$ as

$$\phi(v) = \sum_i \phi(a_i) \ .$$

Observe that ϕ is additive on concatenation of finite words, ie if $v, w \in A^*$ are finite then we can define a concatenation $v \circ w$ (all the actions in v followed by the actions in w), and $\phi(v \circ w) = \phi(v) + \phi(w)$. A word u is called a *prefix* of a word w if there is a word v such that $w = u \circ v$. Such a prefix is *proper* if v contains at least one action. A word $a_1, a_2, \ldots, a_k$ is called a *computation* if $\sigma(a_i) = \pi(a_{i+1})$ for all $i = 1, \ldots, k-1$. Such a computation is called a *history* if $\pi(a_1) = is$. The *final state* of a finite history $a_1, a_2, \ldots, a_k$ is $\sigma(a_k)$. Such a history is called *feasible* if its final state is the goal g. A feasible history is *optimal* if its cost is the minimum over all feasible histories.

We can thus deduce from (1) that *every prefix of an optimal history is itself optimal* in the following sense. Let w^* denote an optimal history, and consider any finite proper prefix u^* of w^*. Then for any history u with the same final state as u^*, $\phi(u^*) \leq \phi(u)$.

Markov processes have the property that there is a total order on actions. Concurrent systems are systems where there is no global clock, and thus no total ordering on actions is generally possible. We can only assume a partial ordering. Because we are interested in considering the shortest path problem for concurrent systems, we wish to use a class of models which captures this behaviour. Petri Nets (PNs) have been shown to be a useful set of models for concurrent systems (see eg [7], [8]). Our first step in generalising MDPs to concurrent systems will be to derive a PN model for an MDP.

A Petri Net is a tuple $\mathcal{P} = (P, T, F, M_0)$ where

- P is a set of *places*,
- T is a set of *transitions*,
- $F \subseteq \{P \times T\} \cup \{T \times P\}$ is a set of (directed) *arcs*, and
- $M_0 \subseteq P$ is the *initial marking*.

[3] The conventional formalism for Bellman's equation uses the notion of an optimal cost to go from a state, rather than the optimal cost to arrive in a state. The two expressions of optimality are equivalent.We adopt the second approach as it is more appropriate for developing the optimality principle we seek for concurrent systems.

A place p is said to be *marked* is there is a token held in p. The system evolves by moving tokens between places. The initial marking M_0 is the set of places initially holding a token. The semantics of a PN are as follows. Consider a transition t. Let $\pi(t) = \{p \in P : \exists\ (p,t) \in F\}$ denote the set of *predecessors* to t. If every place in $\pi(t)$ is marked, we say that t is enabled. An enabled transition *may* fire, in which case, a token is removed from each place in $\pi(t)$, and a token is placed in each place in $\sigma(t) = \{p \in P : \exists\ (t,p) \in F\}$ (the set of *successors* to t). If a transition t fires when the system has marking M, the system marking becomes $M' = (M \setminus \pi(t)) \cup \sigma(t)$. We write $M \xrightarrow{t} M'$. [4] A marking M is *reachable* from M_0 if there is a sequence of firings $t_1, t_2, \ldots, t_k$ such that $M_0 \xrightarrow{t_1} \cdots \xrightarrow{t_k} M$. The sequence $t_1, \ldots, t_k$ is known as an *occurrence sequence*.

Now, let's establish a PN representation for an MDP. We have the following associations specified by an isomorphism ℓ :

- Each state $s \in S$ corresponds to a place $\ell(s) \in P$,
- Each action in $a \in A$ corresponds to a transition $\ell(a) \in T$,
- For each action $a \in A$, there are arcs $(\ell(\pi(a)), \ell(a))$ and $(\ell(a), \ell(\sigma(a)))$ in F,
- $M_0 = \ell(is)$.

We also have a unique place $\ell(g)$ specified by the goal state g. The mappings π, σ and ϕ can be used consistently in the PN, observing that for a PN representation of an MDP, $\pi(t)$ and $\sigma(t)$ each have exactly one element for any transition $t \in T$. It can be easily verified that the goal $\ell(g)$ is reachable from M_0 if and only if the MDP has a shortest path.

The method we propose for determining the optimal policy using the PN representation of the MDP, is based on PN unfoldings [6]. An unfolding is another PN with specific properties that make it possible to determine all possible sequences of transition firings which take the initial marking to the goal marking, and the associated costs. We shall address unfoldings in section 4.

3 PN Representations for Products of MDPs

In this section, we define a concurrent system model based on the idea of products on MDPs. This approach is motivated by the ideas of [6]. Let $\mathcal{M}_i = (S_i, A_i, is_i, g_i, \pi_i, \sigma_i, \phi_i),\ i = 1, \ldots, n$ be MDPs, then we define the product $\mathbf{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n, \mathbf{A}\}$, where $\mathbf{A}$ is a set of *global actions*. The set $\mathbf{A}$ is a subset of

$$(A_1 \cup \epsilon) \times \cdots \times (A_n \cup \epsilon) \setminus \{\epsilon, \ldots, \epsilon\}\ ,$$

where ϵ is a special action which means "do nothing". We call $\mathcal{M}_i$, the i-th *component* of the product $\mathbf{M}$. Global actions have the effect of synchronising

[4] This is considered in the *multiset* framework where a particular place p appears once for each token present in p. We will focus on so-called *one-safe* PNs where the maximum number of tokens in a place is one, so standard set theory suffices.

MDPs through "shared" actions, ie those global actions with more that one non-ϵ component. We say that a component $\mathcal{M}_i$ *participates* in a global action $\mathbf{a} = (a_1, \ldots, a_n)$ if $a_i \neq \epsilon$. We remark that products are inherently partially ordered systems. For the product $\mathbf{M}$ we define the global state space $\mathbf{S} = S_1 \times \cdots \times S_n$, and the global initial state $is = (is_1, \ldots, is_n)$ and the global goal is $g = (g_1, \ldots, g_n)$.

Let $\mathbf{a} = (a_1, \ldots, a_n)$, then the cost of $\mathbf{a}$ is given by

$$\phi(\mathbf{a}) = \sum_{i=1}^{n} \phi_i(a_i) \ ,$$

where $\phi_i(\epsilon) = 0$ for all $i = 1, \ldots, n$. Observe that $\phi(\mathbf{a}) > 0$ for all $\mathbf{a} \in \mathbf{A}$. Let $\mathbf{A}^*$ denote the set of all sequences of global transitions, then we can extend ϕ to $\mathbf{A}^*$ in the same way as described in section 2. The shortest path problem for $\mathbf{M}$ involves finding a sequence of global actions taking the global state from is to g which results in the minimum cost.

PMDPs could be solved in the conventional way by using the so-called *interleaved representation* of the product [6]. This involves applying dynamic programming to the large MDP constructed by enumerating the global state space $\mathbf{S}$. However, such a method is intrinsically inefficient (exponential in the number of components in the product) because it places an unnatural total ordering on events which includes all interleavings of concurrent events. We shall return to this issue in section 4.

We can map products of MDPs to a PN in a way that preserves any concurrency in the system as follows. Let ℓ_i denote the isomorphism which takes component MDP $\mathcal{M}_i$ to its PN representation $\mathcal{P}_i$. Then define a PN $\mathcal{P} = (P, T, F, M_0)$ by

- $P = \cup_{i=1}^{n} \{\ell_i(s_i) : s_i \in S_i\}$,
- for each $\mathbf{a} \in \mathbf{A}$ there is a unique $t \in T$ which we label $\ell(\mathbf{a})$,
- Let $\mathbf{a} \in \mathbf{A}$, then $(\ell_i(s_i), \ell(\mathbf{a})) \in F \Leftrightarrow \exists\ i \ni s_i = \pi_i(a_i) \wedge a_i \neq \epsilon$, and $(\ell(\mathbf{a}), \ell_i(s_i)) \in F \Leftrightarrow \exists\ i \ni s_i = \sigma_i(a_i) \wedge a_i \neq \epsilon$,
- $M_0 = \cup_{i=1}^{n} \ell_i(is_i)$.

Here ℓ is defined on global states by $\ell(\mathbf{s}) = (\ell_1(s_1), \ell_2(s_2), \ldots, \ell_n(s_n))$. We also identify a global goal $\ell(g) = (\ell_1(g_1), \ell_2(g_2), \ldots, \ell_n(g_n))$. We can show that $\ell(g)$ is reachable from M_0 if and only if there is a (global) shortest path.

4 PN Unfoldings

In this section, we will introduce the concept of a branching process of the PN representation of a product of MDPs and thence the unfolding. We'll then show how unfoldings can be used to determine shortest paths. For notational clarity, in the sequel, we shall drop the explicit use of the isomorphism ℓ when the context is clear.

Let $\mathcal{P}$ be a PN representation of a product. A branching process is a special type of PN obtained from $\mathcal{P}$ which has a tree-like structure. In branching

processes, places are called *conditions* and transitions are called *events*. The set of branching processes of $\mathcal{P}$ is constructed as follows. There is a homomorphism ψ defined on the set of PNs such that :

(a) The PN consisting of the initially marked places $\overline{M_0}$ with $\psi(\overline{M_0}) = M_0$, and no events is a branching process of $\mathcal{P}$,
(b) Let $\mathcal{N}$ be a branching process of $\mathcal{P}$ such that there is some reachable marking M in $\mathcal{N}$ with $\psi(M)$ enabling some transition t in $\mathcal{P}$. Let e be an event labelled by $t = \psi(e)$ and $M_1 \subseteq M$ be such that $\psi(M_1) = \pi(t)$. The set of all (e, M_1) is called the set of *possible extensions* to $\mathcal{N}$. Then the PN formed by adding the event e and all conditions c with $\psi(c) \in \sigma(t)$, along with the arcs (e, c) and $\{(m, e) : m \in M_1\}$ to $\mathcal{N}$ is also a branching process of $\mathcal{P}$.
(c) The union of any set of branching processes is also a branching process.

The union of all branching processes is known as the *unfolding* and any of its branching processes is a *prefix* of the unfolding. In general, we seek a *finite* prefix of the unfolding which has all information needed to solve the shortest path problem.

There are two specific technical issues associated with determining an unfolding. Firstly, we need to determine which events should be added to the unfolding as it is progressively computed according to the above process. Secondly, we need to determine when we can stop extending the unfolding after the addition of some new event. We do this by constructing a certain type of ordering on events which is based on the cost function ϕ associated with each event.

Firstly, we need to introduce some additional technicalities. We firstly consider the unfolding $\mathcal{U}$ of the PN representation of a single MDP. We shall define the cost associated with an event e by the cost of its label, ie $\phi(e) = \phi(\psi(e))$ (with obvious abuse of notation). If there is a directed path from an event e to another event e' in $\mathcal{U}$, we say $e < e'$. This is an irreflexive partial order on the events on $\mathcal{U}$ called the *dependency ordering*. Let e be an event in $\mathcal{U}$, and let $e_1, \ldots, e_m$ denote an occurrence sequence with $e_m = e$. The *history* of an event e is the history $\psi(e_1), \ldots, \psi(e_m)$ and is denoted by $H(e)$. The final state of $H(e)$ written $St(e)$ is the state reached after execution of e ie $\sigma(e)$. A history $H(e)$ is called feasible if $St(e) = g$, and then e is called a terminal event. A feasible history is optimal if it has the minimum cost over all such feasible histories. A fundamental property for the single component case [6] is that $H(e) = H(e') \Leftrightarrow e = e'$. This is because every event has a unique predecessor condition.

Let $u, v \in A^*$ be words, then we define a partial order $\prec_\phi$ by $u \prec_\phi v \Leftrightarrow \phi(u) < \phi(v)$. Evidently, $\prec_\phi$ refines the prefix order ie if u is a proper prefix of v, then $u \prec_\phi v$ since costs are positive. We can define the reflexive order $\preceq_\phi$ by saying $u \preceq_\phi v$ if either $u \prec_\phi v$ or $u = v$. Since two words $u \neq v$ can have the same cost, $\preceq_\phi$ is not total. We can thus define an partial ordering on events according to the cost of their histories (as words). It is important for this partial order to refine the prefix order as we want to be able to formulate a principle of optimality that is essentially "causal", ie based on the idea of testing possible extensions for optimality. This is in the spirit of (1).

The unfolding algorithm ERV-Fly [3],[4] proceeds by retaining a queue of all possible extensions to the current branching process. Possible extensions in the queue are ordered according to the cost of their history. Those possible extensions leading to the same state but at a higher cost are discarded. The algorithm terminates when we add an event which corresponds to a transition having the goal state as its successor. Provided the queue ordering ensures that all events on an optimal history are removed from the queue before a terminating event of a non-optimal history, then the unfolding algorithm is guaranteed to find an optimal history. This is proven in more generality in [3],[4] (theorem 4.2.1). This approach embodies a concise characterisation of optimality in the single component case, as theorem 1 states.

Theorem 1. *Let $H(e^*)$ be an optimal history, and let $H(e_0^*)$ be any proper prefix of $H(e^*)$. Then for any history h with the same final state as $H(e_0^*)$, $\phi(H(e_0^*)) \leq \phi(h)$.*

Proof. If there was a history h with $\phi(h) < \phi(H(e_0^*))$, we could form the feasible history $h \circ v^*$, where $H(e^*) = H(e_0^*) \circ v^*$, and $\phi(h \circ v^*) < \phi(H(e^*))$ contradicting optimality of $H(e^*)$.

Comments

1. Theorem 1 is an identical statement to (1) but is in a form that we can naturally generalise to the multicomponent case.
2. This theorem motivates part of the ERV-Fly algorithm [3],[4]. If we have two possible extensions with different events e, e', with the same final state $\sigma(e) = \sigma(e')$, then we add that event with lowest cost history, and we need no longer extend the unfolding from the other event.

4.1 Multiple Components

Let's now address the case of a product of MDPs. In this case, there is no obvious direct analogy to Bellmans's principle, and we seek to develop such an analogy. The main difficulty is that events are not uniquely defined by their histories, so we can't define an ordering on the events in the set of possible extensions based on their histories. However, the notion of the independence of (global) actions is useful for overcoming this difficulty. Two global actions in the PN representation of a product $\mathbf{M}$ are independent if no component of $\mathbf{M}$ participates in both of them. Two words $\mathbf{w}, \mathbf{w}'$ are said to be *1-equivalent* if (i) $\mathbf{w} = \mathbf{w}'$, or (ii) there are independent actions $\mathbf{a}, \mathbf{a}'$ and two words $\mathbf{u}, \mathbf{v}$ such that $\mathbf{w} = \mathbf{uaa'v}$ and $\mathbf{w}' = \mathbf{ua'av}$. We define an equivalence relation $\equiv$ on $\mathbf{A}^*$ as the transitive closure of the 1-equivalence relation. It can be easily shown that if $\mathbf{h}$ is a history of $\mathbf{A}$, then so is every word $\mathbf{w} \equiv \mathbf{h}$.

Importantly, from the viewpoint of optimisation, equivalent words have identical cost as theorem 2 shows.

Theorem 2. *Let $\mathbf{u}, \mathbf{v} \in \mathbf{A}^*$, then $\mathbf{u} \equiv \mathbf{v} \Rightarrow \phi(\mathbf{u}) = \phi(\mathbf{v})$).*

Proof. If $\mathbf{u} \equiv \mathbf{v}$ there is a sequence of permutations P of the actions in $\mathbf{u}$, so that $P(\mathbf{u}) = \mathbf{v}$. Since permutations in the order of actions does not effect the cost of a word, $\phi(P(\mathbf{u})) = \phi(\mathbf{u}) = \phi(\mathbf{v})$. □

Definition 1. *A Mazurkiewicz trace (a trace) for a product* $\mathbf{M}$ *is an equivalence class of words under* $\equiv$*. We write* $[\mathbf{w}]$ *for a trace, and* $[\mathbf{A}^*]$ *for the set of all traces.*

From theorem 2, we can unambiguously define the cost of a trace by extending $\phi : [\mathbf{A}^*] \to \mathbb{R}^+ \cup \{\infty\}$. This allows us to extend the partial order $\prec_\phi$ to traces.

The concatenation of two traces $[\mathbf{u}]$ and $[\mathbf{v}]$, written $[\mathbf{u}] \circ [\mathbf{v}]$ is defined to be the trace $[\mathbf{u} \circ \mathbf{v}]$. We say $[\mathbf{u}]$ is a prefix of $[\mathbf{w}]$ if there is a trace $[\mathbf{v}]$ such that $[\mathbf{w}] = [\mathbf{u}] \circ [\mathbf{v}]$. It is easy to show that the prefix relation is a partial order.

Lemma 1. *The partial order* $\prec_\phi$ *defined on* $[\mathbf{A}^*] \times [\mathbf{A}^*]$ *refines the prefix ordering on traces.*

Proof. Suppose $[\mathbf{w}] = [\mathbf{u}] \circ [\mathbf{v}] = [\mathbf{u} \circ \mathbf{v}]$, then $\phi([\mathbf{w}]) = \phi(\mathbf{w}) = \phi(\mathbf{u}) + \phi(\mathbf{v}) > \phi(\mathbf{u}) \Rightarrow \phi(\mathbf{u}) < \phi(\mathbf{w}) \Leftrightarrow \mathbf{u} \prec_\phi \mathbf{w} \Leftrightarrow [\mathbf{u}] \prec_\phi [\mathbf{w}]$. □

We now introduce the notion of a *configuration* of a branching process. Firstly, any two nodes x, y (condition or event) in a branching process are *causally related* if either $x \leq y$ or $y \leq x$ where $\leq$ is the reflexive dependency ordering. Two nodes are *in conflict* if there is a condition b so that x and y can be reached from b by exiting b on different paths. Two nodes are *concurrent* if they are neither causally related nor in conflict.

A set of events C of a branching process is a configuration if it is both causally closed (ie $e \in C \wedge e' < e \Rightarrow e' \in C$) and conflict-free.

A *realisation* of a set E of events is an occurrence sequence in which each event of E appears exactly once, and no other events appear. Every configuration has at least one realisation. It can be shown that every realisation of a finite configuration C leads to the same marking (see eg [6], proposition 3.18). We call this marking the *global state* $\mathbf{St}(C)$. Given a configuration C, suppose there is configuration $C_0 \subset C$, and a set of events $E \neq \emptyset$ disjoint from C_0 with $C = C_0 \cup E$. We say that C_0 is a *proper prefix* of C and that E is an *extension* of C_0, and write $C = C_0 \circ E$.

We can now define the history of a configuration C as a word $\mathbf{a_1} \ldots \mathbf{a_n}$ such that $e_1 \ldots e_n$, with $\psi(e_i) = \mathbf{a}_i, i = 1, \ldots, n$, is a realisation of C. We denote the set of histories of C by $\mathbf{H}(C)$. A history $\mathbf{H}(C)$ is called *feasible* if $\mathbf{St}(C) = \mathbf{g}$. The cost of a history is its cost defined as a trace. A feasible history is *optimal* if its cost is minimum over all feasible histories. The cost of a configuration is the cost of its history.

The following results are proven in [6] (proposition 4.28). (a) Let C_1 and C_2 be configurations then $C_1 = C_2 \Leftrightarrow \mathbf{H}(C_1) = \mathbf{H}(C_2)$. (b) Let C be a configuration, then $\mathbf{H}(C)$ is a trace. These results are important because they allow us to characterise configurations by their histories and that every history in $\mathbf{H}(C)$ has the same cost, and leads to the same reachable marking. This again leads to an optimality principle.

Theorem 3. *Let C^* denote an optimal configuration and suppose that C_0^* is a proper prefix of C^*. Then for all configurations C_0 with $St(C_0) = St(C_0^*)$, $\phi(C_0^*) \leq \phi(C_0)$.*

Proof. Write $C^* = C_0^* \circ E$ where E is an extension of C_0^*. Then $\mathbf{H}(C^*) = \mathbf{H}(C_0^*) \circ [e]$ for some $e \in \mathbf{A}^*$. Let $\mathbf{H}(C^*) = [c^*]$ and $\mathbf{H}(C_0^*) = [c_0^*]$, since histories of configurations are traces, then $\phi([c^*]) = \phi([c_0^*]) + \phi([e])$. Suppose there was a configuration C_0 with the same final state as C_0^* and with $\phi(C_0) < \phi(C_0^*)$, then with $\mathbf{H}(C_0) = [c_0]$, we have $\phi([c_0] \circ [e]) = \phi([c_0]) + \phi([e]) < \phi([c_0^*]) + \phi([e]) = \phi([c^*])$ which contradicts the optimality of C^*. □

Comments

1. It may seem that the expression of optimality in terms of configurations might be restrictive in the sense that the result doesn't apply in generality to all PMDPs. However the specific property that configurations represent the so-called "true" concurrency semantics of the PMDPs means that theorem 3 is indeed general.
2. Theorem 3 is a necessary condition to be satisfied for any algorithm which yields an optimal solution and is a part of the ERV-Fly algorithm [3],[4]. It is conjectured that the principle might also lead to other algorithms for determining optimal configurations in a similar way that Bellman's principle does. This is a subject of ongoing work. Of course, not all optimisation algorithms that exploit concurrency will necessarily be *derived* from theorem 3, since, in particular, it is based on Petri Nets and unfoldings, however, in some equivalent sense, any truly concurrent optimisation algorithm for PMDPs must satisfy something equivalent to theorem 3.

5 Conclusion

This paper has derived a principle for optimality for a new class of concurrent decision processes called products of Markov Decision Processes (PMDPs). This class of processes is motivated by the concurrent system models studied in [6]. The principle reduces to the standard Bellman's principle of optimality applying to the single MDP case. The principle is embodied in an existing algorithm for concurrent system optimisation called ERV-Fly [3],[4], although our results show it is a necessary part of any optimisation algorithm for PMDPs. Due to the fact that ERV-Fly can be applied to more general Petri Nets (see [3],[4]), we would also conjecture that a similar principle applies to general Petri Net models. This is, in part, supported by the properties of so-called adequate orderings on configurations and generalisations thereof. We also believe that the principle might lead to the development of new optimisation algorithms for concurrent systems in a similar way that Bellman's principle motivated a number of different optimisation algorithms for MDPs such as value iteration, policy iteration, and hybrids of these two methods.

In other ongoing work, we are considering the suitability of the above approach for systems with probabilistic outcomes. These systems are already included in

MDP models but it can be difficult to extend this approach to concurrent systems (see eg [8], [9]). However, the restricted nature of PMDPs which still retain a notion of local state in each component may permit a solution.

Acknowledgements

The authors acknowledge the support of National ICT Australia and the Defence Science and Technology Organisation Australia through the Dynamic Planning, Optimisation and Learning Project. We also thank Sylvie Thiébaux, Patrik Haslum and Jussi Rintanen for useful discussions.

References

1. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 1 and 2. Athena Scientific, Belmont (2001)
2. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann, San Francisco (2004)
3. Hickmott, S.L.: Directed Unfolding: Reachability Analysis of Concurrent Systems and Applications to Automated Planning, Ph.D. Thesis, School of Electrical and Electronic Engineering, The University of Adelaide (2008)
4. Hickmott, S., Rintanen, J., Thiébaux, S., White, L.B.: Planning via Petri Net Unfolding. In: Proc. 20th Int. Joint Conf. on Artificial Intelligence, India, pp. 1904–1911 (2007)
5. Bonet, B., Haslum, P., Hickmott, S., Thiébaux, S.: Directed Unfolding of Petri Nets. In: Workshop on Unfolding and Partial Order Techniques. In 28th Int. Conf. of Application and Theory of Petri Nets and other Models of Concurrency, Poland (2007)
6. Esparza, J., Heljanko, K.: Unfoldings: A Partial Order Approach to Model Checking. Springer, Berlin (2008)
7. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE 77(4), 541–580 (1989)
8. Benveniste, A., Fabre, E., Haar, S.: Markov Nets: Probabilistic Models for Distributed and Concurrent Systems, Rapport de Recherche INRIA RR-4253 (September 2001)
9. Haar, S.: Probabilistic Cluster Unfoldings. Fundamenta Informaticae 53, 3-4, 281–314 (2002)

Partial Order Hierarchical Reinforcement Learning

Bernhard Hengst

Neville Roach Research Laboratory, NICTA, Sydney Australia
School of Computer Science, University of NSW, Sydney, Australia

Abstract. In this paper the notion of a partial-order plan is extended to task-hierarchies. We introduce the concept of a partial-order task-hierarchy that decomposes a problem using multi-tasking actions. We go further and show how a problem can be automatically decomposed into a partial-order task-hierarchy, and solved using hierarchical reinforcement learning. The problem structure determines the reduction in memory requirements and learning time.

1 Introduction

Russel and Norvig [1] note that "Fortunately, the real world is a largely benign place where subgoals tend to be nearly independent". Polya [2] exploits this real-world property, suggesting that a good problem solving heuristics is to ask "Can you decompose the problem and recombine its elements in some new manner?" The decomposition of a problem is usually left to a human designer and a challenge in machine learning is to automate this divide-and-conquer technique.

On a related theme it has often been stated that the complexity we encounter in natural environments takes the form of hierarchy and that hierarchy is one of the central structural schemes that the architecture of complexity uses [3,4]. Hierarchical reinforcement learning (HRL) uses hierarchy to represent and solve problems more efficiently (see [5] for a survey). One HRL approach, MAXQ [6], explicitly uses a task-hierarchy.

Several researchers have looked at automating the decomposition of a reinforcement learning problem into subtasks and recombining their policies to solve the original problem [7,8,9,10,11]. HEXQ [11] is a decomposition algorithm that automatically builds a MAXQ-like task-hierarchy by uncovering structure in problems from state variables. It decomposes a problem by ordering the state variables by their frequency-of-change, constructing a task-hierarchy with one level per variable.

This paper introduces partial-order task-hierarchies using multi-tasking actions that execute several independent subtasks in arbitrary order. The partial-order generalisation of task-hierarchies can represent hierarchical reinforcement problems more succinctly and speed up learning time. We address shortcomings in HEXQ by examining several state partitions of the problem at the same time and by building a more compact partial-order task-hierarchy representation.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 138–149, 2008.

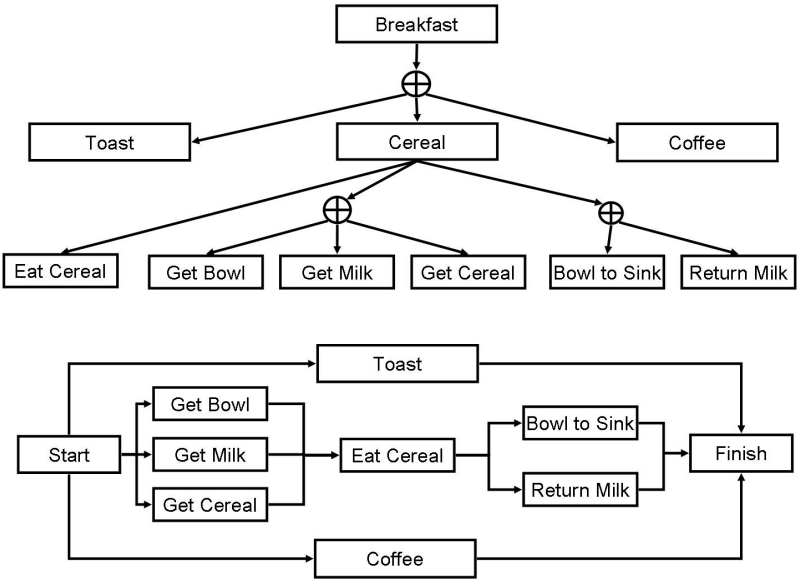


Fig. 1. Top: A partial-order task-hierarchy for a breakfast task. Bottom: The optimal partial-order plan.

We assume the reader is familiar with reinforcement learning [12] and hierarchical reinforcement learning [5]. In the rest of this paper we introduce partial-order task-hierarchies. We describe how a two-level partial-order task-hierarchy is constructed using a simple one-room problem to motivate the discussion. Finally, we show a diverse set of results, using several room mazes, and demonstrate savings in learning time and memory requirements for a larger "breakfast" task.

2 Partial-Order Task-Hierarchies

In a task-hierarchy each action invokes one subtask. In a *partial-order* task-hierarchy we allow actions to invoke several independent subtasks. We call these actions *multi-tasking* actions, because their objective is to complete multiple subtasks before terminating. Multi-tasking actions can be interpreted as committing simultaneously to multiple subgoals, but they are only able to pursue one subgoal at each time-step.

An algorithm that can represent a problem using multiple actions without specifying their order of execution is a partial-order planner [1]. Multi-tasking actions are non-deterministic in this way and can be used to extend the notion of a task-hierarchy to a partial-order task-hierarchy. We indicate multi-tasking actions in a partial-order task-hierarchy graph by the symbol $\bigoplus$ and coalesce edges from the parent task to multiple child subtasks, signifying that all the child subtasks need to complete and terminate before returning control to the parent.

Figure 1 (top) shows an example of a partial-order task-hierarchy for an agent that plans to consume cereal, toast and coffee in some arbitrary order for breakfast. The cereal subtask is further divided into (1) a simple action to eat the cereal, (2) a multi-tasking abstract action to co-locate a bowl from the cupboard, milk from the fridge and cereal from the pantry, and (3) another multi-tasking action to return the milk to the fridge and to take the bowl to the sink.

A partial-order task-hierarchy has the expressive power to represent multiple partial-order plans. For example, in the breakfast task, the order of the three actions available to the cereal subtask, and indeed whether they are needed at all, will be learnt by the hierarchical reinforcement learner. One of the possible partial-order plans, the optimal one, is shown in the bottom of Figure 1.

2.1 The One-Room Problem

We use reinforcement learning (RL) to represent these type of problems. RL is formalised as a Markov decision problem MDP $< S, A, T, R >$ where S is a finite set of states, A a finite set of actions, $T : S \times A \times S \to [0, 1]$ a stochastic transition function and $R : S \times A \times \mathbb{R} \to [0, 1]$ a stochastic reward function. We assume we are provided with states represented as a tuple of variables $(s^1, s^2, ..., s^i, ..., s^n)$. A *policy* is a function that specifies which action to take in any state. We use the action-value function $Q(s, a)$ as the expected undiscounted sum of rewards to termination, starting in state s, taking action a and following a specified policy thereafter. The reader is referred to [12] for further background on RL. In hierarchical reinforcement learning (HRL) parent task state actions can invoke subtasks [5]. Since a subtask can execute for multiple time-steps the parent problem becomes a semi-MDP.

We will use the simple one-room problem in Figure 2 as a running example. In this problem an agent starts at random in a 10 x 10 grid-world room. Its aim is to leave the room by the shortest path. The state is the position of the agent perceived as coordinates (x, y). Although x and y take integer labels they are not assumed to have any numeric meaning. In each state the agent can choose from four stochastic actions - move one step to the North, East, South or West. Actions move the agent in the intended direction 80% of the time, but 20% of the time the agent stays where it is. The reward at each step is -1.

Solving this problem in a straightforward manner using reinforcement learning requires a table of Q values of size $|S||A| = 100 * 4 = 400$. However, the agent could learn to move in the x and y directions independently. The only dependence is at $(2, 9)$ where a move to the North reaches the goal. This results in the decomposition shown by the partial-order task-hierarchy in Figure 2 (right). The root subtask consists of 1 abstract state representing the room. The root multi-tasking action invokes two subtasks, one for navigating North-South, the other East-West. We terminate the multi-tasking action when the North-South subtask state is 9 and the East-West subtask state is 2 followed by the action to move North. The number of Q values required to represent this partial-order task-hierarchy by a hierarchal reinforcement learner is reduced from 400 to only 81 values as will be shown later.

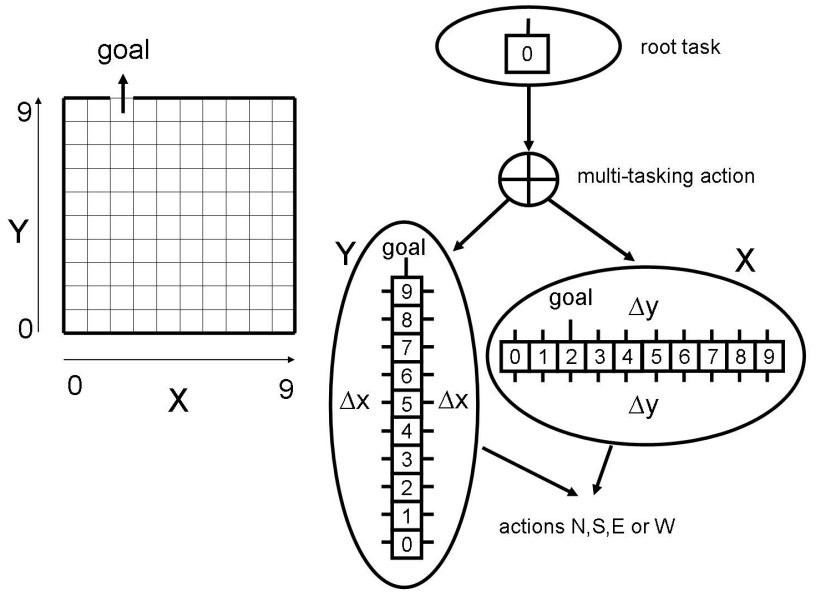


Fig. 2. The one-room problem and its decomposition using a partial-order task-hierarchy with one multi-tasking action

3 Constructing Partial-Order Task-Hierarchies

We will show how our hierarchical reinforcement learner constructs and solves partial-order task-hierarchies taking a closely related approach to HEXQ [11]. HEXQ does not "fail" on any of the tasks in this paper but it can be inefficient because each variable in the state description is forced to add another level to the task-hierarchy. Multi-tasking actions overcome this issue.

Multi-tasking actions implement a similar function to the AND decomposition in StateCharts [13] that capture the property that the system state must be simultaneously in each subtask being multi-tasked. The system state is the orthogonal product of the state variables describing the subtasks. When an action in one of the subtasks does not affect other subtasks it represents a kind of independence which we exploit to decompose MDPs extending the idea of StateChart AND decomposition to stochastic actions.

To discover partial-order task-hierarchies we process all variables in the state tuple concurrently to construct the first level in the hierarchy. For each variable we find the regions and region exits as in HEXQ [11]. In this paper we restrict ourselves to two level hierarchies, but the construction can be extended to multiple levels.

The root task state is also described by a tuple of variables, but now they identify the regions the system state is in at the level below. The root task state is therefore more abstract and factors out detail represented in the child subtask regions. If the problem does not have any structure, regions will be singleton

states and there will be no computational gain in this approach. Once regions and region exits have been identified for all state variables we can set about constructing a set of multi-tasking actions for the root task.

The original task is solved by solving the top level semi-MDP in the abstract state-space using the multi-tasking actions. To solve the semi-MDP the HRL has to commit to a linearization of the multiple tasks. We arbitrarily use the state-tuple variable order, but one can imagine if this approach is used as an approximation, the commitment to the order of processing may be governed by a higher level context. We are afforded the same advantages in the flexibility of execution as for partial-order plans.

3.1 Regions and Exits

We now look at the construction in more detail. A *model* (the transition and reward function) is discovered for each state variable s^i using a random exploration policy as for HEXQ [11]. For each variable we store the transition function as a series of transitions from one state to the next for each action. We also tally the frequency as we experience the transition multiple times and accumulate the rewards to allow us to estimate the transition distribution and expected reward functions.

As in HEXQ an *exit* is a state-action pair used to signify that the transition or reward function cannot be well defined because, for example, there is state aliasing in the projected state space. We declare a transition an exit whenever another variable $s^l, i \neq l$ changes value or the problem terminates. For consistency, problem termination is represented as a boolean goal variable changing value from false to true. Importantly, and in contrast to HEXQ, we also store the set of variables s^l that change value for each exit.

Our hope is that the variables s^i show some independence and that we can discover independent models over at least part of their state-action space. The state-space for each variable is partitioned into regions (i.e. independent model subspaces) using the same criteria as for HEXQ . A *region* is a block of the partitioned state-space ensuring that all exit-states are reachable from all entry states.

As an example, the ovals marked X and Y in Figure 2 show the two projected state-spaces for the variables x and y. In this problem each projected state-space is just one region. Exits are indicated by short lines emanating from each of the 10 states. For the Y region the exits may change the x values and state $y = 9$ may additionally exit to the goal. Similarly, the X region has exits from all states that may change the y variable values.

3.2 Multi-tasking Actions

Multi-tasking actions are automatically generated by examining the exit information from all the regions. They are created by considering all possible combinations of exits for those regions that comprise each abstract root-task state. Multi-tasking actions are represented as a tuple $(s^1, ..., s^n, a)$ indicating the exit

state s^i each subtask should reach before executing the exit action a. The procedure for creating multi-tasking actions is as follows:

- for each abstract state in the root task we identify the unique region associated with each variable s^i. This is straightforward because of the root task abstract state is a tuple of variables indexing these regions (see Section 3)
- we generate all possible combinations of exits, one from each identified region. Region exits are available from the procedure described in Subsection 3.1
- a multi-tasking action is created whenever all the exits in a combination:
  1. have the same exit-action, and
  2. change the same variables in the set of changed variables - the s^l in Subsection 3.1

To justify the first criteria above, we note that we require one action to terminate each multi-tasked subtask to return control to the parent task. Only one action can be performed at a time by the agent and therefore the exit-action must be common to all exits.

The system is simultaneously in all subtasks invoked by a multi-tasking action. When the multi-tasking action terminates it simultaneously terminates all subtasks. A valid multi-tasking action must therefore potentially change the same set of variables, justifying the second criteria. Together these criteria eliminate a large number of combinations.

As for HEXQ, we need to define and solve the MDP associate with each region in the multi-tasking action with the goal of reaching the exit-state and executing the exit-action.

Let's illustrate the construction using the one-room example. The two regions X and Y have 21 and 20 exits respectively as shown in Figure 2. Exit combinations include $\{(x = 4, a = North), (y = 9, a = North)\}$. This combination does not qualify for producing a multi-tasking action because the exits fail to change the same variable s^l. The goal variable is changed by $(y = 9, a = North)$, but not by $(x = 4, a = North)$.

In this problem it turns out that the only exit combination that meets the two criteria for multi-tasking actions is $\{(x = 2, a = North), (y = 9, a = North)\}$ producing only one multi-tasking action $(x = 2, y = 9, a = North)$. Note that in this case they both have exit-action $a = North$ and they both change the same variables, i.e. x, y and the goal.

The X region MDP takes the agent to $x = 2$ and the Y region MDP takes the agent to $y = 9$. The multi-tasking action is completed by executing action $North$. The number of Q values for the root task is $|S||A| = 1 * 1 = 1$ and $|S||A| = 10 * 4 = 40$ for each of the child subtasks, making a total of only 81 vales required to store the one-room task. In comparison HEXQ requires a total of 610 Q values.

The total reward for a multi-tasking action is the cumulative reward from associated subtasks because subtasks are independent and executed serially. As in HEXQ, policies for problems with deterministic actions are optimal and hierarchically optimal for stochastic actions. Hierarchical optimality stems from

having single exit subtasks ensuring that the policies are the best possible given the constraints of the hierarchy [11].

4 Results

We now use several mazes and a larger breakfast task to demonstrate the versatility of automatic partial-order HRL.

4.1 Mazes

The six mazes in Figure 3 are solved using partial-order HRL. In each case we assume a similar problem formulation to the one-room problem. Mazes (a) to

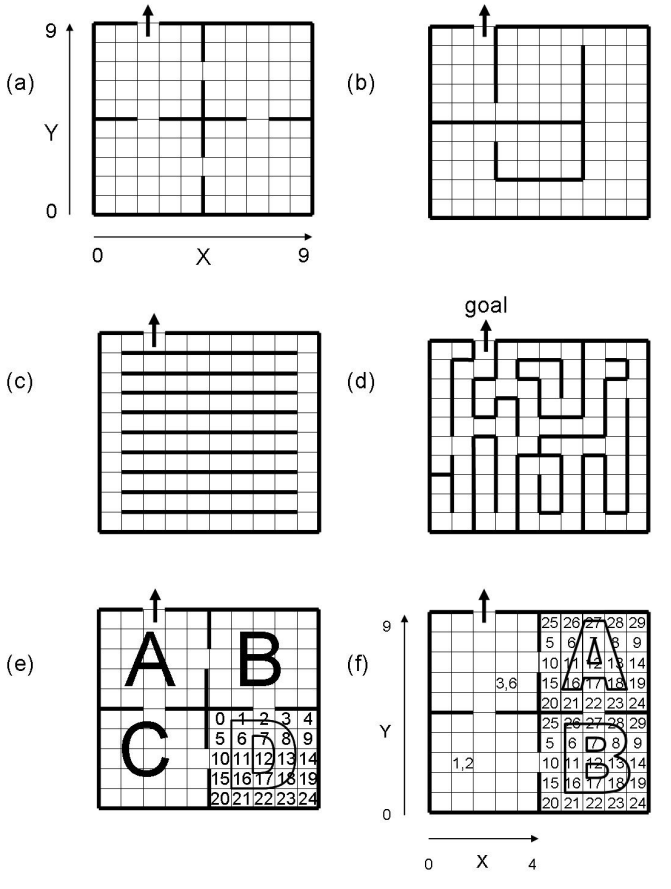


Fig. 3. Mazes used for Partial-Order HRL

Table 1. Summary statistics for the partial-order task-hierarchical decomposition for the various mazes. $|S|$ is the number of abstract states discovered, $\overline{A/S}$ the average number of multi-tasking actions per abstract state generated, $\sum |S||A|$ shows the memory requirements to represent the abstract problem, $\sum |S||A|\%$ the size of the abstract subtask as a % of the original problem size and Tot% is the decomposed problem size as a % of the original problem.

Maze	$\|S\|$	$\overline{A/S}$	$\sum \|S\|\|A\|$	$\sum \|S\|\|A\|\%$	Tot%
(a)	4	10.3	41	10.25	110.3
(b)	9	9.0	81	20.25	92.3
(c)	10	3.7	37	9.250	49.3
(d)	90	3.4	305	76.25	98.3
(e)	4	4.0	16	4.000	110.3
(f)	4	6.8	27	6.750	158.8
Fig 2	1	1.0	1	0.250	20.3

(d) represent the state-space with a coordinate like attribute pair (x, y). Maze (e) is similar to the others except that X now takes on 25 different values to represent the position in each of the rooms and Y has 4 values to identify the room. In maze (f) we mix the two representations from mazes (a) and (e). The left half of the maze is described by coordinates and the right half, by room and position-in-room variables.

We are careful to ensure that state values are not aliased as this would result in a partially observable problem. In each case the order of the variables defining the state vector is arbitrary and any order will give the same results.

The maze in Figure 3 (a) is a four-room problem. This maze generates two regions for each of the variables X and Y as shown in Figure 4. Interestingly the root task that is generated has four abstract states representing the four rooms in the domain. This demonstrates an ability of the algorithm to represent the problem at an abstract level to reflect the structure. There are 11 multi-tasking actions generated for the top-left-room state and 10 for the other three abstract room states. Most of these multi-tasking actions bump the agent into the walls and do not lead the agent out of the rooms. These effects are learnt as the top-level semi-MDP is solved.

Maze (b) decomposes into three regions per variable and hence 9 abstract root level states. Agents starting in the center region of the maze need to follow a spiral like path to reach the goal and learn to cut corners in the lower part of the maze to minimise the distance to the goal.

Maze (c) has one region for the X variable, 10 regions for the Y variable and an average of 3.7 multi-tasking actions per abstract state.

Maze (d) illustrates a more complex situation where it might appear at first that abstraction is not possible. The algorithm found the two-state region, $[y = 1, y = 2]$, and reduced the number of multi-tasking actions for most abstract states due to the constraints in the maze, resulting in a slight reduction in overall memory requirements and the abstract problem reduced to 76% of the original.

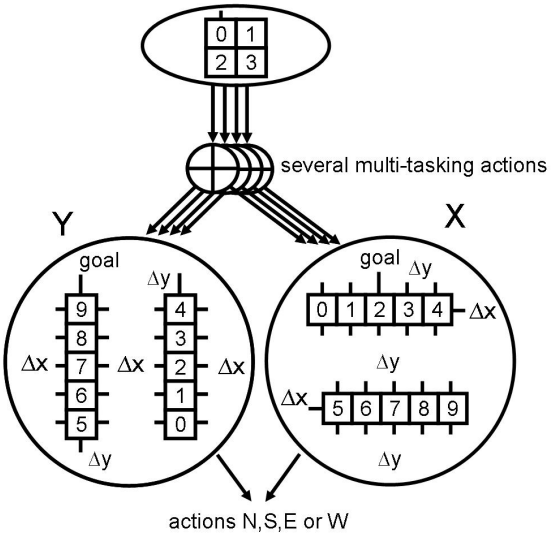


Fig. 4. The automatic decomposition of the four-room problem of maze (a) in Figure 3

Mazes (e) and (f) each use different representations for the base-level states and each still automatically decomposes to an abstract problem with four states representing the four rooms. While the number of abstract states is the same and they refer to the same rooms, the subtasks invoked by their multi-tasking actions vary. In contrast to maze (a), maze (e) in Figure 3 generates 1 region for the X variable and 4 regions for the Y variable. At the abstract level the Figure 3 (e) agent would explore a possible bottom-left room exit to the west, whereas the maze (a) agent would not. The latter already represents the impossibility of a West exit at $x = 0$, while the former knows it may be able to exit West at $x = 10$, but still needs to learn in which rooms.

Table 1 summarises the abstraction results for the room mazes. Memory requirements and problem sizes are measured in terms of the number of table entries required to store the value function. In each case we can find an abstract representation of the problem that is smaller than the original. Some of the mazes require more total storage after decomposition when subtasks are included, but we must remember that in practice problems will be larger with the cost of subtask storage amortised over many more contexts. For example, in maze (e) with four rooms the total partial-order task-hierarchy memory requirements exceed that of the original problem. The total memory requirements for n rooms of size 5 by 5 is $100n$ for the original problem and only $400 + 8n$ for one decomposed with a partial-order task-hierarchy. We start reducing the total problem size when $n > 4$ rooms.

4.2 The Breakfast Task

We demonstrate a decomposition of a breakfast task that uses three variables and stochastic actions. A robot must learn to co-locate a bowl, milk and cereal on a table, eat the cereal, replace the milk in the refrigerator and move the bowl to the sink to be washed up. We represent the kitchen as a 2D 100 location grid-world in which the refrigerator, cereal cupboard, table and sink have separate locations. The state-space is defined by a vector giving the location of the bowl (100 locations), the milk (100 locations) and the cereal (101 locations, one being in the stomach). There are 12 primitive actions allowing each of the objects to be moved Up, Down, Left, Right, and one action to eat the cereal. The actions moving each object are stochastic in the sense that the location of an object at the next step is 80% as intended and 20% as if moved in another direction with equal likelihood. The reward per action is -1 and 1000 for successful completion.

Partial-order HRL constructs the task-hierarchy for the cereal subtask in Figure 1 (top) and finds the abstract policy that is equivalent to the center part of the partial-order plan, Figure 1 (bottom). There are only two abstract states with one multi-tasking action each. The reinforcement learning specification for the original representation of the problem requires a table size of 13,130,000 compared to a partial-order task-hierarchy decomposition of 6,515. Figure 5 shows

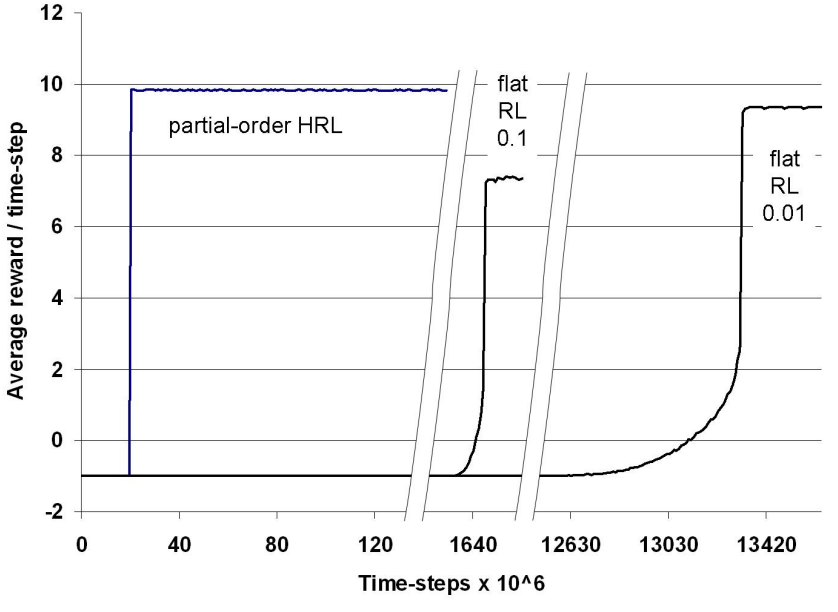


Fig. 5. Breakfast task performance: Partial-order HRL outperforms the "flat" RL by orders of magnitude in learning time and in the level of performance. Two learning rates of 0.1 and 0.01 were used for the "flat" RL.

the improvement in performance. The results including the time to explore and build the subtasks for the partial-order hierarchical reinforcement learner.

5 Future Work and Conclusion

Our contention is that this type of decomposition is common in real world tasks and that robots need to decompose the environment in this way to effectively solve everyday chores. For example in assembly operations the order of construction of components parts themselves may not matter. When writing a paper the order of preparation of the figures is not important, etc.

Future work envisages automatic decomposition when subtasks are interdependent with conflicting or cooperative goals to achieve near optimal performance. The multi-tasking action $\bigoplus$ will then need to arbitrate the decision as to which action to execute within a subtask so as to best compromise between subtask goals. Related research that implements a run-time choice between (possibly) conflicting child subtasks includes W-Learning [14], modular Q-Learning [15], Q-Decomposition [16] and Co-articulation [17]. The challenge is to extend the automatic discovery of multi-tasking actions to include conflicting subtask choices and continuing (infinite horizon) problems.

Task-hierarchies can be further extended to include abstract concurrent actions $\bigotimes$ by projecting both state and action vectors and modelling the state-action spaces $s^i \times a^j$ for all i, j. The approach would entail abstracting subsets of variables and actions and searching for independent regions that can be executed concurrently. A key difference between a concurrent and a multitasking action is that the former implies parallel execution of primitive actions and the latter does not.

The contributions of this paper include the introduction of partial-order task-hierarchies and their use in automatic problem decomposition and efficient solution of MDPs using a hierarchical reinforcement learning approach. Given that in the real world subtasks tend to be nearly independent, this representation is anticipated to help scale reinforcement learning significantly.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River (1995)
2. Polya, G.: How to Solve It: A New Aspect of Mathematical Model. Princeton University Press, Princeton (1945)

3. Turchin, V.F.: The Phenomenon of Science. Columbia University Press (1977)
4. Simon, H.A.: The Sciences of the Artificial, 3rd edn. MIT Press, Cambridge (1996)
5. Barto, A., Mahadevan, S.: Recent advances in hiearchical reinforcement learning. Special Issue on Reinforcement Learning, Discrete Event Systems Journal 13, 41–77 (2003)
6. Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research 13, 227–303 (2000)
7. Thrun, S., Schwartz, A.: Finding structure in reinforcement learning. In: Tesauro, G., Touretzky, D., Leen, T. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 7. MIT Press, Cambridge (1995)
8. Digney, B.L.: Emergent hiearchical control structures: Learning reactive hierarchical relationships in reinforcement environments. In: From Animals to Animats 4: Proceedings of the fourth international conference on simulation of adaptive behaviour, pp. 363–372 (1996)
9. McGovern, A., Sutton, R.S.: Macro-actions in reinforcement learning: An empirical analysis. Amherst technical report 98-70, University of Massachusetts (1998)
10. Şimşek, O., Barto, A.G.: Using relative novelty to identify useful temporal abstractions in reinforcement learning. In: Proceedings of theTwenty-First International Conference on Machine Learning, ICML 2004 (2004)
11. Hengst, B.: Discovering hierarchy in reinforcement learning with HEXQ. In: Sammut, C., Hoffmann, A. (eds.) Proceedings of the Nineteenth International Conference on Machine Learning, pp. 243–250. Morgan Kaufmann, San Francisco (2002)
12. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
13. Harel, D.: Statecharts: A visual formalism for complex systems. Science of Computer Programming 8, 231–274 (1987)
14. Humphrys, M.: Action selection methods using reinforcement learning. In: Maes, P., Mataric, M., Meyer, J.A., Pollack, J., Wilson, S.W. (eds.) From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, pp. 135–144. MIT Press, Bradford Books (1996)
15. Karlsson, J.: Learning to Solve Multiple Goals. PhD thesis, University of Rochester (1997)
16. Russell, S., Zimdars, A.: Q-decomposition for reinforcement learning agents. In: Proc. ICML 2003, Washington, DC (2003)
17. Rohanimanesh, K., Mahadevan, S.: Coarticulation: an approach for generating concurrent plans in markov decision processes. In: ICML 2005: Proceedings of the 22nd international conference on Machine learning, pp. 720–727. ACM Press, New York (2005)

Optimal Global Path Planning in Time Varying Environments Based on a Cost Evaluation Function

Om K. Gupta and Ray A. Jarvis

Department of Electrical and Computer Systems Engineering
Monash University, Clayton, VIC 3800, Australia
{Om.Gupta,Ray.Jarvis}@eng.monash.edu.au

Abstract. This paper describes a unique and optimal method for real-time global path planning and collision avoidance for navigation of a mobile robot in complex time varying environments. Occupancy based 3D grid map and Gaussian distribution model based obstacle prediction are employed to represent the dynamic environment. Path planning and obstacle avoidance are performed by applying a cost-evaluation function on time-space Distance Transforms to uniquely produce the optimal path at the time of planning. Experimental results are presented verifying the effectiveness and versatility of the algorithm in both predictable and imperfectly predictable time varying environments.

1 Introduction

Path planning in an unknown environment or partially unknown environment with dynamic obstacles has always been a challenging problem and the core of many research efforts. The motion of a moving entity in such environment is quite often unpredictable and can not be perfectly modelled.

Many approaches and algorithms have been proposed for such time varying environments and implemented on various robots. However, only few of these discuss scenarios with obstacle speed at that of humans or higher, and with multiple moving entities. The methods based on potential field approach such as Virtual Force Field (VFF) [1] applies attractive force field towards goal and repulsive force fields away from obstacles to calculate the path. Other methods based on curvature velocity method (CVM) [2] and dynamic window approach [3] take account of the trajectory property of the motion to avoid colliding with obstacles. These methods only work perfectly in static or very slow time varying environments. To ensure collision-free movements for the mobile robots in the presence of other moving objects, one approach is to take obstacle motion estimation as the first step, before an obstacle avoidance algorithm is employed as suggested by Fiorini in Velocity Obstacle approach [4]. Other methods proposed were Hidden Markov Model (HMM) [5] and Visual Servoing [6]. However, these are computationally very expensive and performance cannot be obtained in real-time and also fail to deliver an optimal path in many situations, especially when the obstacle velocities are higher than the robot.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 150–156, 2008.

Our method not only ensures real-time performance for time varying environments, but also finds the shortest route to its destination. Several examples are presented in complex scenarios for the optimal approach to avoid obstacles while moving towards the goal. Section 2 presents a brief background study leading to the development of our approach, which is described in Section 3. A detailed analysis of the results and the performance of the method based on experiments in a 3D virtual world is presented in Section 4.

2 Previous Work

2.1 Distance Transform (DT)

Rosenfeld and Pfaltz [7] introduced the DT algorithm for image processing in 2D binary image array mainly for analysis of shape of a blob. Jarvis [8] extended it for path planning for a mobile robot in the space of the robot's environment. The basic approach of the method is to propagate distance in an incremental manner out from specified goals, flowing around obstacles, until all of the free space has a distance from the nearest goal parameter associated with it [9]. Moving down the steepest descent path from any free-space cell in this distance transformed space generates the least-cost (in distance terms) path to the nearest goal. The DT method is quite popular for its simplicity and robustness. It can also be easily adapted to many situations such as, indoor or outdoor, known or unknown, static or time-varying obstacle field environments.

2.2 Occupancy Grid (OG)

Autonomous robots must be able to learn and maintain models/maps of their environments or/and obstacles for successful navigation. A 3D Grid-based map is used to represent the configuration space of a robot projected into an x-y plane with evenly spaced grid cells, known as an Occupancy Grid. Occupancy Grids were originally proposed by Elfes and Moravec [10] and have a value attached to each grid cell that measures the probabilistic belief that a cell is occupied based on sensor readings. This is one of the simplest and widely used models which perfectly fit to our environment model.

2.3 Visibility Based Path Planning

Marzouqi [11] proposed a new visibility-sensitive path planning for covert robotic navigation. The aim was to minimize the robot's exposure to hostile sentries. The approach depends on estimating a cost value at each free-space location that presents the risk of being seen by any sentry. Based on the DT algorithm methodology, the minimum visibility-distance cost to a goal is calculated at each cell in the grid-based environment map. Moving along the steepest descent trajectory from any starting point generates an optimal covert path to a goal. Our method minimises a path cost made up of accumulated weighted mixtures of distance and risk of collision, thus providing efficient and low risk trajectories. The approach is explained in depth in the following section.

3 Our Approach

3.1 Environment Model

Time varying environment is represented using a three-dimensional grid based map, where two of the dimensions are used to represent the 2D physical space, and time providing an extra dimension as the time/space model. The occupancy map of the system includes permanent obstacles and places which are inaccessible to robot, such as, wall, furniture, hole in a ground, etc.

3.2 Obstacle Model and Cost Evaluation

Obstacle Modelling concerns the capturing and integration of volumetric occupancy and visual data from the environment to define just where the robot might move and possibly also aid in its localisation. Our obstacle model includes all the dynamic obstacles or initially unknown static obstacles. The model confirms the occupancy of a cell by exhibiting the probability of it for being occupied by an obstacle depending on the sensor readings. The probability of each cell in the occupancy grid (which are within the field-of-view of sensor) being occupied by an obstacle, P_{obs}, after a sensor reading s, is updated by applying Equation (1).

$$P_{obs}(s) = (1 - \alpha)P_{obs}(s - 1) + \alpha C(s) \tag{1}$$

where, α is the confidence constant indicating probability of a cell being an obstacle given the sensor reading to be positive for obstacle, $C(s) = 1$ for a cell if sensor reads positive for obstacle, and $C(s) = 0$ for a negative sensor reading.

Given two consecutive readings for an obstacle location, the velocity of the obstacle is determined. If the obstacles are perfectly predictable, the velocities can be used to predict the exact position of the obstacles and robot itself in the future. For an imperfectly predictable environment, the velocity is used to construct the probability model for the obstacle in third dimension, using the Gaussian distribution model to determine the position of obstacle in future frames. As time progresses, the mean position of Gaussian distribution, i.e., Gaussian peak decreases and drifts, and standard deviation increases. This is expected because of the increase in uncertainty for obstacle location in the future. The amount of the drift vector is determined from the predicted velocity of the obstacle.

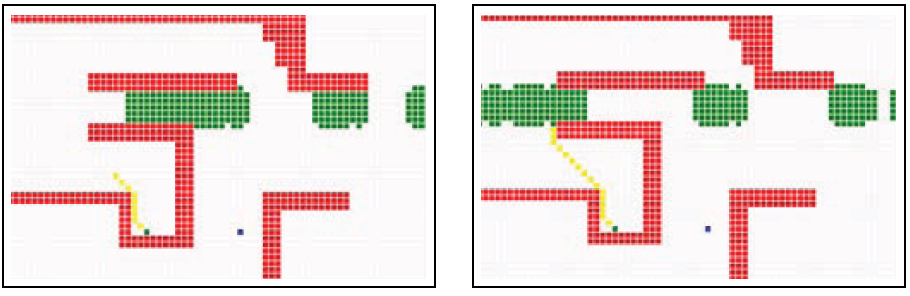
The obstacle cost $\Theta(c)$, i.e., the weight of the occupancy can easily be evaluated from the probability calculation multiplied by a tuneable constant, δ.

$$\Theta(c) = \delta P_{obs}(c) \tag{2}$$

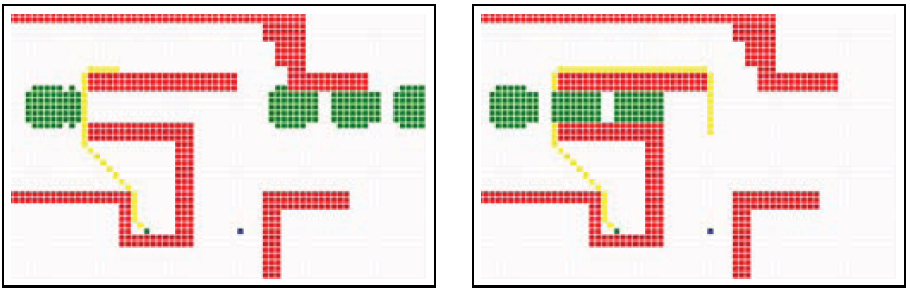
Safer path can be obtained by choosing higher δ value relative to distance costs, which is described in the following subsection.

3.3 Distance Cost

A simple and intuitive cost structure $\zeta(c)$ is used for distance cost in a 3D grid based on Manhattan Distance. Standing still for one time interval is assigned a



(a) Time Frame, $t + 9\Delta t$ **(b)** Time Frame, $t + 16\Delta t$



(c) Time Frame, $t + 31\Delta t$ **(d)** Time Frame, $t + 71\Delta t$

Fig. 1. A scenario where longer path is preferred by the algorithm

unit cost. Moving forward, backward or sideways is assigned a cost of two units, and moving diagonally a three cost units. Using this cost structure, the distance cost $\Phi(c)_t$ of cell c for time layer t was calculated as in Equation (3):

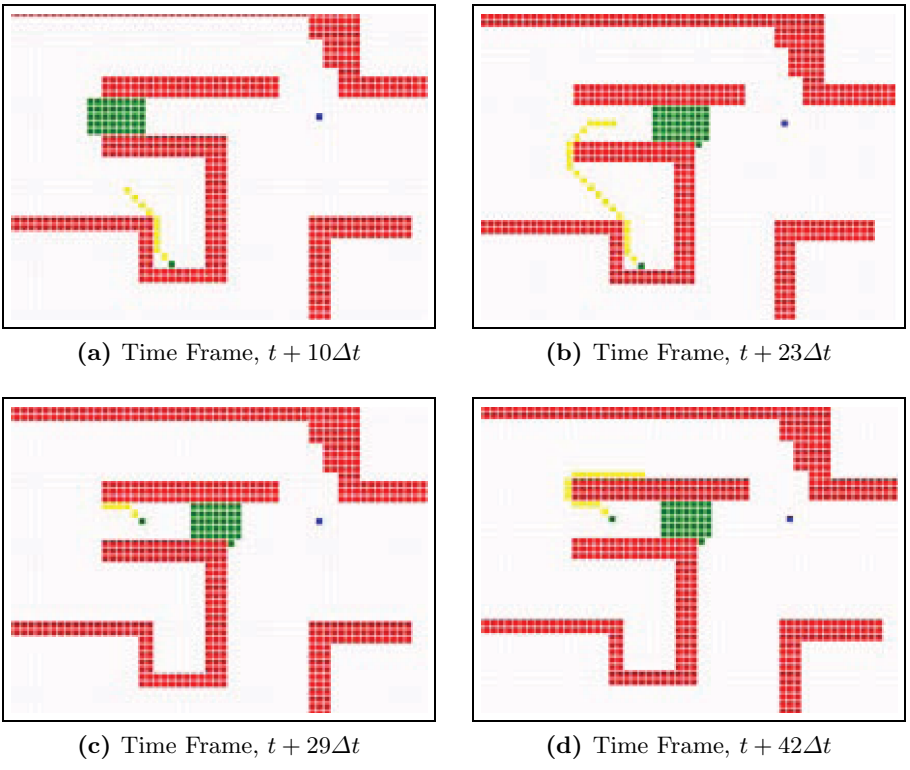
$$\Phi(c)_t = min_{i=1}^{9}[\Phi(n(i))_{t-1} + \zeta(n(i))_{t-1}] \tag{3}$$

where, $\zeta(n(i))_{t-1}$ is the cost from the cost structure between time layer $t-1$ and t from a neighbour $n(i)$ to the current cell c. The distance cost is calculated from destination growing outwards. The irreversibility of time property is exploited in assigning this DT based cost by allowing a single pass raster scanning only along one direction of time.

3.4 Path Planning Algorithm

An important advantage offered by the DT is its ability to accept additional cost factors other than the distance. Once the obstacle cost and the distance cost are calculated, the total cost $\Psi(c)_t$ for a cell c can be obtained for path planning from Equation (4). If no moving obstacles are present in the map, then the distance cost can also be used as the total cost.

$$\Psi(c)_t = min_{i=1}^{9}[\Psi(n(i))_{t-1} + \zeta(n(i))_{t-1} + \Theta(c)] \tag{4}$$



(a) Time Frame, $t + 10\Delta t$ **(b)** Time Frame, $t + 23\Delta t$

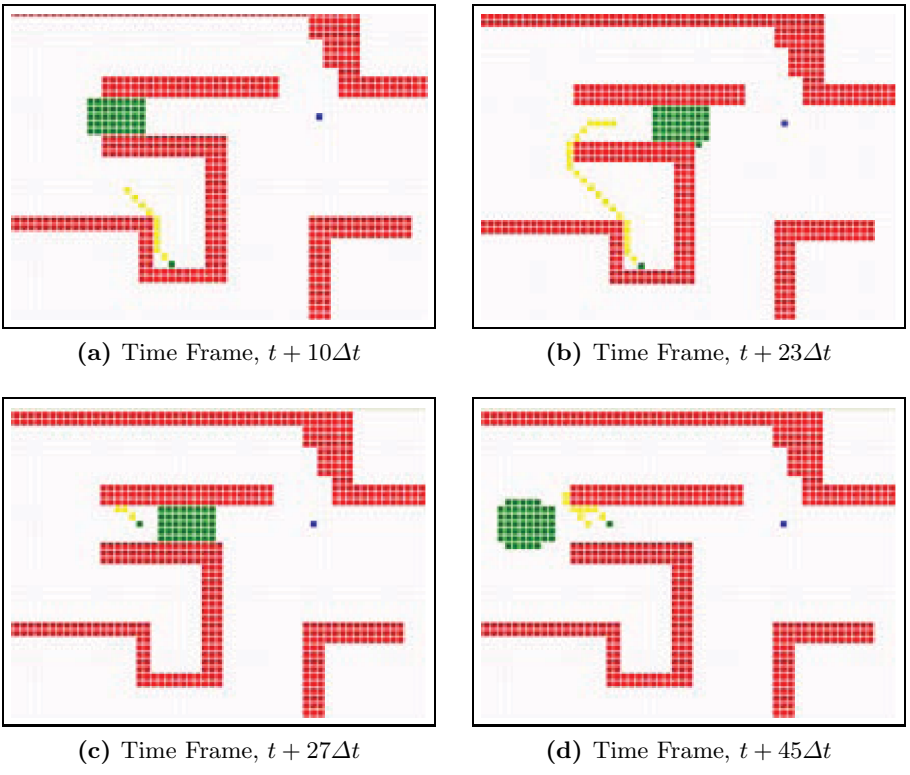
(c) Time Frame, $t + 29\Delta t$ **(d)** Time Frame, $t + 42\Delta t$

Fig. 2. A scenario where the moving entity walking in front of the robot suddenly stops blocking the path

Once again, the algorithm starts from destination growing outward as in distance cost. It can be seen from Equation (4) that the total cost can directly be computed from obstacle cost, avoiding an extra step of distance cost calculation. Finally, path for a robot is obtained by following the steepest descent path from the robot's position towards the destination as explained earlier. The generated path can be tuned to provide a balance between length and risk of the path by adjusting the tuneable constant δ.

4 Experimental Results

The technique described in the previous section was tested rigorously for a diverse range of scenarios. Experiments were conducted in a virtual 3D simulated world with settings for obstacles of different shapes, sizes, locations and velocities. Robot position was assumed to be a point, of about the size of a cell in the map and obstacles were dilated by appropriate amount in order to avoid the collision with the robot. It was assumed that the localisation of the robot position, obstacles and moving entities were accurately estimated in the experiments.



(a) Time Frame, $t + 10\Delta t$ **(b)** Time Frame, $t + 23\Delta t$

(c) Time Frame, $t + 27\Delta t$ **(d)** Time Frame, $t + 45\Delta t$

Fig. 3. A scenario where the moving entity walking in front of the robot makes a sudden u-turn towards the robot

Figure 1 represents a collection of key snapshots from the simulator at different time slices for a complex scenario with variable speed moving obstacle fields. When the robot encountered the predicted moving obstacles, it waited until the first obstacle passed and then chose to go across before continuing towards its destination. As it demonstrates, in some circumstances choosing a longer path or waiting for a obstacle to pass can be an optimal approach. Our algorithm automatically deals with such situations. The above circumstance can easily be assumed in a common office environment, when the flow of traffic through a corridor is higher.

Figure 2 is a classic case for an unpredictable environment. Imagine a person walking in front of a robot in a narrow hallway suddenly stops, may to look at a painting on the wall or other reasons. In such scenario, the robot recalculates its path from its present location and continues on its newly found optimal path. If no new path is found, robot waits for the person to move out and then continues.

The scenario in Figure 3 is a slight different case from Figure 2 in that the moving entity in front of the robot makes a sudden u-turn towards the robot in a narrow corridor. In such case, the algorithm gives priority to the moving entity

by pulling itself back and giving way to the entity. Once the obstacle is out of the way, the robot continues towards the goal.

5 Conclusion

From the results, it is apparent that the algorithm performs very well for optimal path planning in both predictable and unpredictable time-variable environments. The algorithm is not only simplistic, but also adaptive and flexible to allow different conditions for cost evaluations such as collision-risk factor and path optimality.

The memory usage and computational complexity can be further reduced by integrating the algorithm with topological mapping system and parallelising the computational steps. Because of the versatility of the algorithm, it can easily be extended for many different applications, such as, following or tracking people, covert path planning for moving sentries, indoor and outdoor navigations, which are the open areas of future research work.

References

1. Borensteing, J., Koren, Y.: Real-time Obstacle Avoidance for Fast Mobile Robots. IEEE Transactions on Systems, Man and Cybernetics 19(5), 1179–1187 (1989)
2. Simmons, R.: The Curvature-Velocity Method for Local Obstacle Avoidance. In: Proceedings of International Conference on Robotics and Automation (April 1996)
3. Ogren, P., Leonard, N.E.: A Tractable Convergent Dynamic Window Approach to Obstacle Avoidance. In: Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, EPFL, Lausanne, Switzerland (October 2002)
4. Fiorini, P., Shiller, Z.: Motion Planning in Dynamic Environments Using Velocity Obstacles. The Int. Journal of Robotics Research 17(7), 760–772 (1998)
5. Zhu, Q.: Hidden Markov model for Dynamic Obstacle Avoidance of Mobile Robot Navigation. IEEE Trans. On Robotics and Automation 7(3) (June 1991)
6. Matsumoto, Y., Inaba, M., Inoue, H.: View-based approach to robot navigation. In: Proceeding of IEEWRSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 1702–1708 (2000)
7. Rosenfeld, A., Pfaltz, J.L.: Sequential Operations in Digital Image processing: J.A.C.M. 13(4), 471–494 (Octber 1966)
8. Jarvis, R.A.: Collision-Free Trajectory Planning Using Distance Transofrms. In: Proceedings of Nat. Conf. and Exhibition on Robotics, Melbourne, August 20-24 (1984)
9. Jarvis, R.A.: Distance Transform Based Path Planning for Robot Navigation. In: Zheng, Y.F. (ed.) Recent Trends in Mobile Robots, River Edge, New Jersey, vol. 129, pp. 3–31. World Scientific Publishers, Singapore (1993)
10. Elfes, A.: Sonar-based real-world mapping and navigation. IEEE Journal of Robotics and Automation RA3(3), 249–265 (1987)
11. Marzouqi, M.S., Jarvis, R.A.: New visibility-based path-planning approach for covert robotic navigation. Journal: Robotica (2006)

Using Probabilistic Feature Matching to Understand Spoken Descriptions

Ingrid Zukerman, Enes Makalic, and Michael Niemann

Faculty of Information Technology
Monash University
Clayton, VICTORIA 3800, Australia
{ingrid,enesm,niemann}@csse.monash.edu.au

Abstract. We describe a probabilistic reference disambiguation mechanism developed for a spoken dialogue system mounted on an autonomous robotic agent. Our mechanism performs probabilistic comparisons between features specified in referring expressions (e.g., size and colour) and features of objects in the domain. The results of these comparisons are combined using a function weighted on the basis of the specified features. Our evaluation shows high reference resolution accuracy across a range of spoken referring expressions.

1 Introduction

This paper describes a reference disambiguation mechanism developed for the language interpretation module of a robot-mounted spoken dialogue system. This dialogue system will eventually interact with people in order to enable the robot to assist them in household tasks. Hence, we focus on referring expressions which feature household objects, such as "the small chair" and "the big blue mug". Our mechanism performs probabilistic comparisons between the features stated in a referring expression and the features of candidate objects (e.g., those in the room) in order to determine how well these objects match the specified features. The system currently handles several feature types, including lexical item, colour, size, ownership and location, exhibiting high reference resolution accuracy. The contributions of our mechanism are: (1) probabilistic procedures that perform feature comparisons, and (2) a function that combines the results of these comparisons. These contributions endow our mechanism with the ability to handle imprecise or ambiguous referring expressions. For instance, the expression "the small chair" is ambiguous in the context of a room that contains a big chair, a small armchair and a divan, as none of these objects matches the given description perfectly. Our mechanism ranks these objects according to how well they match the lexical item ('chair') and size ('small') specifications in the utterance. If an ambiguity remains, this ranking supports the formulation of a clarification question.

This paper is organized as follows. Section 2 outlines the interpretation process, and briefly discusses the estimation of the probability of an interpretation. The probabilistic feature comparison process is described in Section 3. Section 4 presents the evaluation of the reference resolution mechanism. Related research and concluding remarks are given in Sections 5 and 6 respectively.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 157–167, 2008.

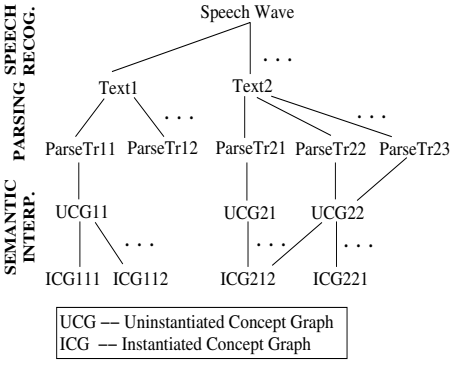


Fig. 1. Stages of the interpretation process

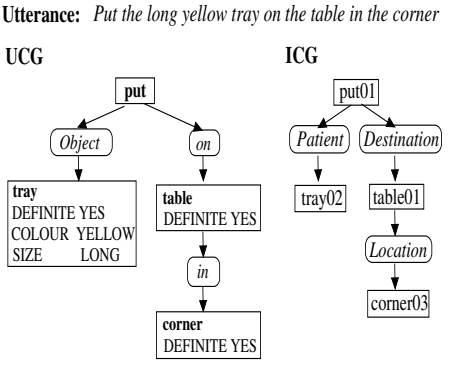


Fig. 2. UCG and ICG for a sample utterance

2 Interpretation Process

Our language interpretation module, called *Scusi?*, processes spoken input in three stages: speech recognition, parsing and semantic interpretation (Figure 1). In the first stage, it runs the Automatic Speech Recognizer (ASR) Microsoft Speech SDK 5.1 to generate candidate texts from a speech signal. Each text is assigned a score that reflects the probability of the words given the speech wave. The second stage applies Charniak's probabilistic parser (`ftp://ftp.cs.brown.edu/pub/nlparser/`) to generate parse trees from the texts. The parser generates up to N ($= 50$) parse trees for each text, associating each parse tree with a probability. During semantic interpretation, parse trees are successively mapped into two representations based on Concept Graphs [1]: first *Uninstantiated Concept Graphs (UCGs)*, and then *Instantiated Concept Graphs (ICGs)*. UCGs are obtained from parse trees deterministically — one parse tree generates one UCG. A UCG represents syntactic information, where the concepts correspond to the words in the parent parse tree, and the relations are derived from syntactic information in the parse tree and prepositions. Each UCG can generate many ICGs. This is done by nominating different instantiated concepts and relations from the system's knowledge base as potential realizations for each concept and relation in a UCG.

Figure 2 illustrates a UCG and an ICG for an utterance containing two referring expressions ("long yellow tray" and "table in the corner"). The *intrinsic* features of an object (e.g., colour and size of the tray) are stored in the UCG node for this object, while *composite* features that involve several objects (e.g., "the *table* in the *corner*") are represented as sub-graphs of the UCG (and then the ICG). This is because intrinsic features can be compared directly to features of objects in the knowledge base, while features that depend on the relationship between several objects require the identification of these objects and the verification of this relationship. In our example, all the tables and corners in the room need to be considered, and the table/corner combination that best matches the given specification is eventually selected. In this paper, we focus on algorithms for selecting objects that match intrinsic specifications (Section 3). The

determination of objects that match composite specifications is done by the procedure that builds ICGs and estimates their overall probability [2].

Our interpretation algorithm applies a sequence of selection-expansion cycles to build a search graph, where each level of the graph corresponds to a stage of the interpretation process (Figure 1). In each cycle, our algorithm selects an option for consideration (speech wave, textual ASR output, parse tree or UCG), and then expands this option to the next level of interpretation. When an option is expanded, a single candidate is returned for this next level. For example, when we expand a UCG, the ICG-generation module returns the next most probable ICG for this UCG. The selection-expansion process is repeated until one of the following happens: all options are fully expanded, a time limit is reached, or a specific number of iterations is performed. At any point after completing an expansion, *Scusi?* can return a list of ranked interpretations (ICGs) with their parent sub-interpretations (text, parse tree(s) and UCG(s)).

2.1 Estimating the Probability of an ICG

Scusi? ranks candidate ICGs according to their probability of being the intended meaning of a spoken utterance. Given a speech signal W and a context $\mathcal{C}$, the probability of an ICG I is represented as follows.

$$\Pr(I|W,\mathcal{C}) \propto \sum_{\Lambda} \Pr(I|U,\mathcal{C}) \cdot \Pr(U|P) \cdot \Pr(P|T) \cdot \Pr(T|W) \tag{1}$$

where U, P and T denote a UCG, parse tree and text respectively. The summation is taken over all possible paths $\Lambda = \{P, U\}$ from a parse tree to the ICG, because a UCG and an ICG can have more than one parent. The ASR and the parser return an estimate of $\Pr(T|W)$ and $\Pr(P|T)$ respectively; and $\Pr(U|P) = 1$, since the process of generating a UCG from a parse tree is deterministic.

The estimation of $\Pr(I|U,\mathcal{C})$ is described in detail in [2]. Here we present the final equation obtained for $\Pr(I|U,\mathcal{C})$, and outline the ideas involved in its calculation.

$$\Pr(I|U,\mathcal{C}) \approx \prod_{k \in I} \Pr(u|k) \Pr(k|k_p, k_{gp}) \Pr(k|\mathcal{C}) \tag{2}$$

where k is an instantiated node in ICG I, u is the corresponding node in UCG U, k_p is the parent node of k, and k_{gp} the grandparent node. For example, *Location* is the parent of `corner03`, and `table01` the grandparent of `corner03` in the ICG in Figure 2.

- $\Pr(u|k)$ is the "match probability" between the features of a node k in ICG I and the corresponding node u in UCG U, e.g., how similar an object in the room is to "the long yellow tray" (Section 3).
- $\Pr(k|k_p, k_{gp})$ is the structural probability of ICG I, where structural information is simplified to node trigrams (e.g., whether the *Location* of `table01` is `corner03`).
- $\Pr(k|\mathcal{C})$ is the probability of a concept in light of the context, which at present includes only domain knowledge, i.e., all instantiated concepts have the same prior. In the future, we propose to estimate these prior probabilities by combining salience scores obtained from dialogue history with visual salience.

3 Probabilistic Feature Comparison

In this section, we describe how *Scusi?* handles three intrinsic features: lexical item, colour and size (other intrinsic features, such as shape and texture, will be handled like colour). For instance, "the big red cup" specifies the features `lexical_item="cup"`, `colour="red"`, and `size="big"`. These features are compared to the features of objects in the domain in order to propose suitable candidates for "the big red cup" when an ICG is created from a parent UCG (semantic interpretation, Section 2).

At present, we make the simplifying assumptions that (1) the robot is co-present with the user and the possible referents of an utterance; and (2) the robot has an unobstructed view of the objects in the room and up-to-date information about these objects (obtained through a scene analysis system [3]). These assumptions obviate the need for planning physical actions, such as moving to get a better view of certain objects or leaving the room to seek objects that better match the given specifications.

Building a list of candidate concepts. For each node u in a UCG U, the task is to construct a list of candidate instantiated concepts $k \in \mathcal{K}$ that are a reasonable match for u ($\mathcal{K}$ is the knowledge base of objects in the domain). This list is built as follows.

1. Estimate $\Pr(u|k)$, the probability of the match between the features of u and those of each candidate instantiated concept k.
2. Rank the candidates in descending order of probability.
3. Filter out the candidates whose probability falls below a certain threshold, and put them in a reserve list (the threshold is feature dependent and determined empirically). This list is accessed if the "A-list" candidates are deemed unsuitable when factor $\Pr(k|k_p, k_{gp})$ in Equation 2 is calculated.

For example, consider a request for a blue mug, assuming the knowledge base contains several mugs, some of which are blue. Firstly, for all the concepts in the knowledge base, we calculate the probability that they could be called 'mug' (e.g., mugs, cups, containers). Next, we estimate the probability that their colour could be considered 'blue'. The candidates are then ranked in descending order of probability, and those whose probability exceeds a threshold are retained in the A-list. In this example, aqua cups stay in the A-list, while red mugs are placed on reserve (sometimes it may be better to offer a red mug than no mug at all).

Estimating the probability of a match. The probability of the match between a node u specified in UCG U and a candidate instantiated concept $k \in \mathcal{K}$ is estimated as follows.

$$\Pr(u|k) = \Pr(\mathbf{u}_{f_1}, \ldots, \mathbf{u}_{f_p} | \mathbf{k}_{f_1}, \ldots, \mathbf{k}_{f_p}) \tag{3}$$

where $(f_1, \ldots, f_p) \in \mathcal{F}$ are the features specified with respect to node u, $\mathcal{F}$ is the set of features allowed in the system, $\mathbf{u}_{f_i}$ is the value of the i-th feature of UCG node u, and $\mathbf{k}_{f_i}$ is the value of this feature for the instantiated concept k.

Assuming that the features of a node are independent, the probability that an instantiated concept k matches the specifications in a UCG node u can be rewritten as

$$\Pr(u|k) = \prod_{i=1}^{p} \Pr(\mathbf{u}_{f_i} | \mathbf{k}_{f_i}) \tag{4}$$

We use a distance function $h : \mathbb{R}^+ \rightarrow [0, 1]$ to map the outcome of a feature match to the probability space. Specifically,

$$\Pr(\mathbf{u}_f|\mathbf{k}_f) = h_f(\mathbf{u}_f, \mathbf{k}_f) \tag{5}$$

In the absence of other information, it is reasonable to perform a linear map, and interpret the resultant values as probabilities. That is, the higher the similarity between requested and instantiated feature values (the shorter the distance between them), the higher the probability of a feature match.

In the following sections, we present the calculation of Equation 5 for the intrinsic features supported by our system (lexical item, colour and size). In agreement with [4,5], lexical item and colour are considered *absolute* features, and size (e.g., big, small, long) is considered a *relative* feature (relative to the size of other candidates).

3.1 Lexical Item

We employ the Leacock and Chodorow similarity measure [6], denoted *LC*, to compute the similarity between the lexical feature of u and that of k (the *LC* measure yielded the best results among those in [7]). The *LC* similarity score, denoted s_{LC}, is obtained from a word-similarity database constructed with the aid of WordNet. This score is converted to a probability by applying the following h_{lex} function.

$$\Pr(\mathbf{u}_{\text{lex}}|\mathbf{k}_{\text{lex}}) = h_{lex}(s_{LC}(\mathbf{u}_{\text{lex}}, \mathbf{k}_{\text{lex}})) = \frac{s_{LC}(\mathbf{u}_{\text{lex}}, \mathbf{k}_{\text{lex}})}{s_{\max}}$$

where $s_{\max}$ is the highest possible *LC* score.

3.2 Colour

We currently make the simplifying assumptions that lighting conditions are good, and that colour is independent of object type and of the colour of surrounding objects. Colour is represented by means of the CIE 1976 (L, a, b) colour model, which has been experimentally shown to be approximately perceptually uniform [8]. L denotes brightness ($L = 0$ for black, and $L = 100$ for white), a represents position between green ($a < 0$) and red ($a > 0$), and b position between blue ($b < 0$) and yellow ($b > 0$). The range of L is $[0, 100]$, and the practical range of a and b is $[-200, 200]$. The probability of a colour match between a UCG concept u and an instantiated concept k is

$$\Pr(\mathbf{u}_{\text{colr}}|\mathbf{k}_{\text{colr}}) = h_{colr}(\mathbf{u}_{\text{colr}}, \mathbf{k}_{\text{colr}}) = 1 - \frac{ED(\mathbf{u}_{\text{colr}}, \mathbf{k}_{\text{colr}})}{d_{\max}}$$

where *ED* is the Euclidean distance between the (L, a, b) coordinates of the colour specified for u and the (L, a, b) coordinates of the colour of k, and $d_{\max}$ is the maximum Euclidean distance between two colours (=574.5). For instance, the (L, a, b) coordinates for blue, azure and royal blue are (29.6, 68.3, -112.1), (98.8,-5.1,-1.8) and (46.8, 17.8, -66.7) respectively, yielding *ED*(blue,royal_blue)= 70.05 and *ED*(blue,azure) = 149.5, with the corresponding probabilities 0.88 and 0.74 (the mapping from colour

name to RGB was obtained from public web sites, e.g., `http://en.wikipedia.org/wiki/List_of_colors`, and a mapping function is used to convert RGB to CIE [8]). Thus, given a request for a blue cup, a royal blue cup has a higher colour match probability than an azure cup. This probability is calculated for the objects that pass the threshold for lexical item (see *Building a list of candidate concepts*).

3.3 Size

Unlike lexical item and colour, size is considered a relative feature, i.e., the probability of a size match between an object $k \in \mathcal{K}$ and a UCG concept u depends on the sizes of all suitable candidate objects in $\mathcal{K}$ (those that exceed the thresholds for lexical and colour comparisons). The highest probability for size match is then assigned to the object that best matches the required size, while the lowest probability is assigned to the object which has the worst match with this size.

This requirement is achieved by the following h_{size} function, which like Kelleher's pixel-based mapping [9], performs a linear mapping between $\mathbf{u}_{\text{size}}$ and $\mathbf{k}_{\text{size}}$.

$$\Pr(\mathbf{u}_{\text{size}}|\mathbf{k}_{\text{size}}) = h_{size}(\mathbf{u}_{\text{size}}, \mathbf{k}_{\text{size}}) = \begin{cases} \dfrac{\alpha \mathbf{k}_{\text{size}}}{\max_i\{\mathbf{k}^i_{\text{size}}\}} & \text{if } \mathbf{u}_{\text{size}} \in \{\text{`large'/`big'/}\ldots\} \\ \dfrac{\alpha \min_i\{\mathbf{k}^i_{\text{size}}\}}{\mathbf{k}_{\text{size}}} & \text{if } \mathbf{u}_{\text{size}} \in \{\text{ `small'/`little'/}\ldots\} \end{cases} \tag{6}$$

where α is a normalizing constant, and $\mathbf{k}^i_{\text{size}}$ is the size of candidate object k^i (this formula is adapted for individual dimensions, e.g., when "a tall mug" is requested).

For example, suppose there are only three cups in the room, $\texttt{cup}_1$, $\texttt{cup}_2$ and $\texttt{cup}_3$, with volumes 0.5, 0.6 and 0.9 dm^3 respectively. The table below shows the probabilities obtained for each cup for the expression "the large cup" and "the small cup" (the highest probability is highlighted).

$\mathbf{k} =$	$\texttt{cup}_1$		$\texttt{cup}_2$		$\texttt{cup}_3$
$\Pr(\text{large cup}\|\mathbf{k}_{\text{size}})$	$\frac{0.5\alpha}{0.9}$	$<$	$\frac{0.6\alpha}{0.9}$	$<$	$\mathbf{\frac{0.9\alpha}{0.9}}$
$\Pr(\text{small cup}\|\mathbf{k}_{\text{size}})$	$\mathbf{\frac{0.5\alpha}{0.5}}$	$>$	$\frac{0.5\alpha}{0.6}$	$>$	$\frac{0.5\alpha}{0.9}$

Although this size-comparison scheme has produced satisfactory results (Section 4), it has the limitation that the size match probability of the instantiated concepts depends on the number of candidates being considered. In the future, we propose to investigate a size scheme based on people's perceptions.

3.4 Combining Feature Scores

People refer to objects by drawing attention to a range of features, e.g., colour, composition and location, which distinguish the target objects from potential distractors. Dale and Reiter [4] found that people often present features that are not strictly necessary to identify an item, and ranked features in the order *type* $\succ$ *absolute adjectives* $\succ$ *relative adjectives*, where colour is an absolute adjective and size is a relative adjective. These findings prompted us to incorporate a weighting scheme into Equation 4, whereby the

features are weighted according to their usage in referring expressions. That is, higher ranking features are assigned a higher weight than lower ranking features.

Since Equation 4 is a product of the match probability for each feature in a referring expression, simply multiplying this probability by a weight w_{f_i} for each feature f_i will not affect the overall probability of the various candidate objects. To overcome this problem, we used the weight of a feature to adjust the range of values of the match probability for this feature. Specifically, given a match probability $\Pr(\mathbf{u}_{f_i}|\mathbf{k}_{f_i})$ and a weight w_{f_i} $(0 < w_{f_i} \leq 1)$ for feature f_i, the adjusted match probability is

$$\Pr'(\mathbf{u}_{f_i}|\mathbf{k}_{f_i}) = \Pr(\mathbf{u}_{f_i}|\mathbf{k}_{f_i}) \times w_{f_i} + \tfrac{1}{2}(1 - w_{f_i}) \quad (7)$$

The effect of this mapping is that features with high weights have a wider range of probabilities (centered on 0.5), and hence a stronger influence on match probability, than features with low weights. For instance, $w_{f_i} = 0.6$ yields $0.2 \leq \Pr'(\mathbf{u}_{f_i}|\mathbf{k}_{f_i}) \leq 0.8$, while $w_{f_i} = 0.8$ yields $0.1 \leq \Pr'(\mathbf{u}_{f_i}|\mathbf{k}_{f_i}) \leq 0.9$. In the future, we intend to investigate a weighting scheme based on exponentiation [10].

4 Evaluation

We performed two experiments to evaluate our system. The first experiment focused on determining the best feature weights, and the second on measuring *Scusi?*'s overall interpretation performance. Both experiments have the following aspects in common. (a) They are based on six "worlds" constructed by us, each comprising 13-26 objects (World 2 is shown in Figure 3); the knowledge base for each world contained only the items in this world; the designation, size and colour of these items were chosen so they had similar features, e.g., a stool may also be called 'seat' or 'chair', and there were objects in different shades of the same colour. (b) Utterances were recorded by one of the authors, as the ASR is speaker dependent, and at present we do not handle features of spontaneous speech. (c) *Scusi?* was set to generate at most 300 sub-interpretations in total (including texts, parse trees, UCGs and ICGs) for each utterance in the test set; this takes 6 seconds from parse tree to full interpretation.

Experiment 1. We composed a set of 7-15 descriptions for each world (e.g., "the long yellow tray"), yielding a total of 56 referring expressions (these expressions were consistent with those generated by our trial subjects in Experiment 2). To establish which objects should be considered the Gold referent, pictures of the six worlds and their accompanying descriptions were shown to two human taggers. The taggers independently identified one or more objects which best corresponded to each description (inter-tagger agreement was $\kappa = 0.86$ [11]). When the taggers disagreed, Gold standards were derived through consensus-based annotation [12], but multiple Gold referents were allowed if several objects in the domain matched a specified object, e.g., "a mug".

Ideally, a machine learning approach should be used to determine the best weight assignment for the features (w_{lex}, w_{colr}, w_{size}) in Equation 7. However, our corpus and domain are too small. Hence, we tried different feature combinations, and report here on the features that yielded the most interesting results.

Fig. 3. Items in World 2

	# Gold refs with prob in top 1	in top 3	Not found	Avg adjust rank (rank)	Avg # refs to Gold (iters)
Experiment 1					
BASELINE	26	47	0	1.33 (1.25)	2.8 (22)
DRW-HIGH	28	51	0	0.91 (0.84)	2.8 (21)
DRW-MED	33	54	0	0.70 (0.63)	2.7 (20)
DRW-LOW	33	50	0	0.82 (0.75)	2.8 (20)
Total	56	56			(300)
Experiment 2					
DRW-MED	49	65	1	0.85 (0.82)	0.88 (18)
Total	75	75			(300)

Fig. 4. *Scusi?*'s performance for Experiment 1 with different feature weights, and for Experiment 2 with best weights

- BASELINE $(1, 1, 1)$.
- DRW – three weighting schemes based on the feature ordering proposed by Dale and Reiter [4] and Wyatt [5]: DRW-HIGH $(1, 0.9, 0.8)$, DRW-MED $(1, 0.8, 0.6)$, and DRW-LOW $(1, 0.67, 0.33)$.

Figure 4 summarizes our results for both experiments. Column 1 shows the weighting scheme. Columns 2 and 3 show how many of the descriptions had Gold referents whose probability was the highest (top 1) or among the three highest (top 3), e.g., for Experiment 1, the baseline scheme yielded 26 referents with the top probability, and 47 referents within the top 3 probabilities. Column 4 indicates the number of cases where Gold referents were not found. The average *adjusted rank* and *rank* of the Gold referent appear in Column 5. The rank of a referent r is its position in a list sorted in descending order of probability (starting from position 0), such that all equiprobable referents are deemed to have the same position. The adjusted rank of a referent r is the mean of the actual positions of all the referents that have the same probability as r. For example, if we have 3 top-ranked equiprobable referents, each has a rank of 0, but an adjusted rank of $\frac{0+2}{2}$. Column 6 indicates the average number of referents created and iterations performed until the Gold referent was found (from a total of 300 iterations).

For this experiment, the ASR had a 20% error rate (the correct text did not have the top score in 20% of the cases). The three weighted schemes outperformed the baseline (the difference in performance is statistically significant with $p < 0.03$ [1]). All the schemes found all the Gold referents, but the rank and adjusted rank of the referents is lower (better) for the weighted schemes, in particular for DRW-MED. In fact, DRW-MED appears to exhibit the best performance, but this result is statistically significant only for DRW-MED versus DRW-HIGH ($p < 0.01$). These results indicate that reference disambiguation performance is improved by adjusting the probabilities of intrinsic features using feature weights which reflect priorities obtained from observations. However, further experiments are required to determine an optimal weighting scheme.

[1] Sample paired t-tests were used for all statistical tests.

Experiment 2. We gathered a corpus of referring expressions by conducting an online survey as follows. We showed pictures of our six worlds, and selected several objects as target referents for each world (for each target, the rest of the objects in the world formed the set of distractors). For example, objects A, E, F, H, K, N and P were the targets for World 2 (Figure 3). Subjects were then asked to identify each target referent using descriptions comprising a lexical item, and optionally colour and size. We logged 178 written referring expressions in total from eight trial subjects unrelated to the project. 80 of these expressions had aspects that are currently not handled by *Scusi?*, such as composite nouns (e.g., "coffee cup"), and attributes indicating composition (e.g., "wooden table") or shape (e.g., "rectangular platter"); and 17 of the remaining expressions were duplicates. This left 81 unique expressions, which were recorded by one of the authors. Six of these utterances were not recognized by the ASR (specifically the words "ramekin" and "biro"), leaving 75 utterances for the evaluation.

The DRW-MED scheme, which yielded the best performance in Experiment 1, was used in Experiment 2. Despite a higher ASR error of 26% for this experiment, *Scusi?*'s performance — summarized in the bottom part of Figure 4 — was comparable to that obtained for DRW-MED and DRW-LOW in Experiment 1. Specifically, the Gold referent was found in the top ranked interpretation in 65% of the cases (compared to 59% in Experiment 1), and in the top-3 interpretations in 87% of the cases (compared to 96% for DRW-MED and 89% for DRW-LOW). Only one Gold referent was not found (a 'pouf' that was referred to as an 'armchair' — these terms have a lexical similarity of 0).

5 Related Research

Much of the research on reference resolution has focused on the construction of expressions that single out a target object from a set of distractors. Dale and Reiter's seminal work [4] proposes several schemes for generating both concise and over-specified referring expressions. They advocate a method for incremental reference resolution that includes unnecessary modifiers, due to its similarity to human behaviour. Wyatt [5] uses Markov decision processes to generate descriptions which place the lowest cognitive load on the addressee; he models the degree of uncertainty about an intended target, and assigns prior probabilities to objects on the basis of their visual salience. Unlike *Scusi?*, Wyatt considers scenes with only a few objects, and does not consider lexical ambiguity (e.g., a mug cannot be referred to as a 'cup' or 'dish'). Siddharthan and Copestake [13] address the lexical ambiguity problem by using antonym and synonym lists from WordNet. However, both of these systems allow only binary comparisons, which precludes the probabilistic ranking of lexical matches.

Methods for understanding referring expressions in dialogue systems are examined in [9,14], among others. Kelleher [9] proposes a reference resolution algorithm that accounts for four attributes: lexical type, colour, size and location, where the score of an object is estimated by a weighted combination of the visual and linguistic salience scores of each attribute. Like in *Scusi?*, the values of the weights are pre-defined and based on observations. However, Kelleher limits the probabilistic comparison of features to size and location, and uses binary comparisons for lexical item and colour. Pfleger *et al.* [14] use modality fusion to combine hypotheses from different analyzers

(linguistic, visual and gesture), choosing as the referent the first object satisfying a 'differentiation criterion'. As a result, their system does not handle situations where more than one object satisfies this criterion.

6 Conclusion

We have offered a probabilistic reference disambiguation mechanism that considers intrinsic features. Our mechanism performs probabilistic comparisons between features specified in referring expressions (specifically lexical item, colour and size) and features of objects in the domain, and combines the results of these comparisons using a weighted function of the features. Our mechanism was evaluated in two experiments, exhibiting good performance, in particular when the features were weighted according to the ordering suggested in [4,5].

In the future, we intend to investigate an alternative size scheme that does not depend on the number of candidates (Section 3.3), and to further investigate feature weighting schemes (Sections 3.4 and 4). In addition, we propose to remove the co-presence and unobstructed-view assumptions (Section 3), which will demand the integration of our feature comparison mechanism with planning procedures.

References

1. Sowa, J.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
2. Zukerman, I., Makalic, E., Niemann, M., George, S.: A probabilistic approach to the interpretation of spoken utterances. In: Ho, T.-B., Zhou, Z.-H. (eds.) PRICAI 2008. LNCS (LNAI), vol. 5351, pp. 581–592. Springer, Heidelberg (2008)
3. Makihara, Y., Takizawa, M., Shirai, I., Miura, J., Shimada, N.: Object recognition supported by user interaction for service robots. In: Proceedings of the 16th International Conference on Pattern Recognition, Quebec, Canada, vol. 3, pp. 561–564 (2002)
4. Dale, R., Reiter, E.: Computational interpretations of the Gricean maxims in the generation of referring expressions. Cognitive Science 18(2), 233–263 (1995)
5. Wyatt, J.: Planning clarification questions to resolve ambiguous references to objects. In: Proceedings of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Edinburgh, Scotland, pp. 16–23 (2005)
6. Leacock, C., Chodorow, M.: Combining local context and WordNet similarity for word sense identification. In: Fellbaum, C. (ed.) WordNet: An Electronic Lexical Database, pp. 265–285. MIT Press, Cambridge (1998)
7. Pedersen, T., Patwardhan, S., Michelizzi, J.: WordNet: Similarity – measuring the relatedness of concepts. In: AAAI 2004 – Proceedings of the 19th National Conference on Artificial Intelligence, San Jose, California, pp. 25–29 (2004)
8. Puzicha, J., Buhmann, J., Rubner, Y., Tomasi, C.: Empirical evaluation of dissimilarity measures for color and texture. In: Proceedings of the 7th IEEE International Conference on Computer Vision, Kerkyra, Greece, vol. 2, pp. 1165–1172 (1999)
9. Kelleher, J.: Attention driven reference resolution in multimodal contexts. Artificial Intelligence Review 25, 21–35 (2006)
10. Potamianos, G., Neti, C.: Stream confidence estimation for audio-visual speech recognition. In: ICSLP 2000 Proceedings, Beijing, China, vol. 3, pp. 746–749 (2000)

11. Carletta, J.: Assessing agreement on classification tasks: The Kappa statistic. Computational Linguistics 22(2), 249–254 (1996)
12. Ang, J., Dhillon, R., Krupski, A., Shriberg, E., Stolcke, A.: Prosody-based automatic detection of annoyance and frustration in human-computer dialog. In: ICSLP 2002 Proceedings, Denver, Colorado, pp. 2037–2040 (2002)
13. Siddharthan, A., Copestake, A.: Generating referring expressions in open domains. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, pp. 407–414 (2004)
14. Pfleger, N., Alexandersson, J., Becker, T.: A robust and generic discourse model for multimodal dialogue. In: Proceedings of the 3rd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Acapulco, Mexico (2003)

Working for Two: A Bidirectional Grammar for a Controlled Natural Language

Rolf Schwitter

Centre for Language Technology
Macquarie University
Sydney NSW 2109, Australia
schwitt@ics.mq.edu.au

Abstract. This paper introduces the controlled natural language PENG Light together with a language processor that is based on a bidirectional grammar. The language processor has the following interesting properties: (a) it translates declarative sentences written in PENG Light into a first-order logic notation (TPTP); (b) it generates declarative sentences in PENG Light taking syntactically annotated TPTP formulas as input; and (c) it translates questions written in PENG Light into (conjunctive) queries in TPTP notation and uses the TPTP representation of the query as a starting point for generating answers in PENG Light. Moreover, the controlled natural language processor can be interfaced directly with an automated reasoner in order to resolve anaphoric references and to answer questions stated in PENG Light.

Keywords: Controlled Natural Languages, Grammar Engineering, Human-Computer Interfaces, Knowledge Representation, Question Answering.

1 Introduction

A controlled natural language is an engineered subset of a natural language with explicit constraints on grammar, lexicon, and style [1]. These constraints usually have the form of writing rules and help to reduce both ambiguity and complexity of full natural language. In general, controlled natural languages fall into two categories: human-oriented and machine-oriented controlled natural languages. Human-oriented controlled natural languages aim at improving text comprehension for human readers while machine-oriented controlled natural languages focus on improving text processability for machines.

During the last ten years, there has been substantial work in the area of machine-oriented controlled natural languages. These controlled natural languages have been designed and used for specification purposes, knowledge acquisition and knowledge representation, and as interface languages to the Semantic Web – among them Attempto Controlled English [2,3], Boeing's Computer-Processable Language [4,5], Common Logic Controlled English [6,7], and PENG Processable English [8,9]. Some machine-oriented controlled natural

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 168–179, 2008.

languages require the user to learn a small number of construction and interpretation rules [3], while other controlled natural languages provide writing support that takes most of the burden of learning and remembering the language from the user [10]. The commercial success of the human-oriented controlled natural language ASD Simplified Technical English [11] suggests that people can learn to work with restricted English and that good authoring tools can drastically reduce the learning curve [12].

In this paper, we will introduce PENG Light and show that this controlled natural language can be processed by a **bidirectional** grammar in contrast to other controlled natural languages. For example, PENG Light's predecessor PENG translates declarative sentences and questions via discourse representation theory [13] into various first-order notations for automated reasoning but not backwards into controlled natural language [9]. We will demonstrate that in the case of PENG Light the same grammar can be used to translate sentences and questions into a first-order notation (that is augmented with syntactic information) and that this notation can be used as a starting point for generating answers to questions in controlled natural language.

2 PENG Light

PENG Light distinguishes between simple sentences and complex sentences. The writing process of these sentences can be supported with the help of a predictive authoring tool [10] that enforces the restrictions of the controlled language.

2.1 Simple Sentences

Simple PENG Light sentences have the following functional structure:

```
subject + predicator + [ complements ] + { adjuncts }
```

The subject has the form of a noun phrase that contains at least a nominal head in form of a noun (= common noun or proper noun). The predicator is realised by a verb that functions as the verbal head of a verb phrase. The existence of complements depends on the verb. An intransitive verb (*1*) does not take any complements, a transitive verb (*2*) takes one complement (a direct object), a ditransitive verb (*3*) takes two complements (a direct object and an indirect object), and a linking verb (*4*) takes a so-called subject complement:

1. *John Miller works.*
2. *John Miller teaches COMP249.*
3. *John Miller sends a letter to Mary.*
4. *John Miller is in the lecture hall.*

For example, in sentence (*2*), the direct object has the form of a noun phrase; in sentence (*3*), the direct object has the form of a noun phrase and the indirect object has the form of a prepositional phrase; and finally in (*4*), the subject

complement has the form of a prepositional phrase. In contrast to complements, adjuncts are always optional and are used in PENG Light to describe the verb in more detail. They have the form of prepositional phrases (*5*) or adverbs (*6*):

5. *John Miller teaches COMP249 on Monday.*
6. *John Miller works exactly.*

Not only verbs but also the nominal head of a noun phrase can be described in more detail in PENG Light: either by adding an adjective (*7*) or a possessive noun (*8*) as a pre-head modifier to the common noun or by adding a variable (*9*) or an *of*-construction (*10*) as a post-head modifier:

7. *The clever professor teaches COMP332 on Monday.*
8. *Mary's father teaches a computing unit in E6A.*
9. *The professor X teaches COMP348 on Monday.*
10. *The father of Mary teaches a computing unit in E6A.*

Note that the translation of the subject noun phrase in (*8*) and (*10*) will result in the same logical representation. Variables are used instead of personal pronouns in order to establish precise anaphoric references within and between sentences and to exclude potential ambiguities (see Section 2.3 for more details).

2.2 Complex Sentences

Complex PENG Light sentences are built from simpler sentences through quantification, negation, subordination and coordination. A special form of complex sentences are conditionals and definitions.

Quantification. PENG Light distinguishes between universal quantification and existential quantification ((*11*) and (*12*)), and qualified cardinality restriction (*13*):

11. *Every professor teaches a unit.*
12. *A professor teaches every unit.*
13. *John Miller teaches* [*exactly* | *at least* | *at most*] *two units.*

There is no scope ambiguity in PENG Light since the textual occurrence of a quantifier determines its scope that extends to the end of a sentence. For example, in sentence (*11*), the universal quantifier has scope over the existential quantifier, and in (*12*), the existential quantifier has scope over the universal one. Note that qualified cardinality restriction – as illustrated in (*13*) – can only occur in complement position in PENG Light but not in subject position.

Negation. PENG Light distinguishes three forms of negation: sentence negation (*14*), noun phrase negation (*15*), and verb phrase negation (*16*) and (*17*):

14. *If is false that a professor teaches COMP225.*
15. *No professor teaches COMP225.*

16. *Every professor does not teach COMP225.*
17. *John Miller is not a professor.*

Sentence negation is realised via the constructor phrase *it is false that*; noun phrase negation is introduced via the universal negative quantifier *no*; and verb phrase negation is realised via the negatives *does not* and *is not*. Note that (*15*) and (*16*) have the same meaning and will result in the same formal representation but will carry different syntactic annotations.

Subordination. Using a relative clause is another option to modify a noun in a noun phrase. In PENG Light, a relative clause is always introduced by a relative pronoun and modifies the immediately preceding noun. Relative clauses trigger so-called filler-gap dependencies since they have a structure where a phrase (or a partial phrase) is missing from its normal position (= gap) and another phrase (= filler) stands for the missing phrase outside the normal position, for example:

18. [*John Miller*]$_{filler}$ *who* [ ]$_{gap}$ *teaches a unit supervises Mary.*
19. [*Mary*]$_{filler}$ *who John Miller supervises* [ ]$_{gap}$ *takes COMP249.*

In the case of a subject relative clause such as in (*18*), the entire noun phrase in subject position is missing, and in the case of a complement relative clause such as in (*19*), the entire complement is missing from its normal position. Note that sentence (*18*) as well as sentence (*19*) can be expressed alternatively with the help of two simple PENG Light sentences as illustrated in (*20*) and (*21*):

20. *John Miller teaches a unit. John Miller supervises Mary.*
21. *John Miller supervises Mary. Mary takes COMP249.*

Relative clauses can also be used in combination with an existential *there*-construction to clarify the scope of the existential quantifier. Instead of (*12*), we can write (*22*):

22. *There is a professor who teaches every unit.*

Coordination. PENG Light distinguishes three forms of coordination: verb phrase coordination (*23*), relative clause coordination (*24*), and modifier coordination in adjunct position (*25*):

23. *Marc does not teach COMP249 and does not supervise Mary.*
24. *John Miller supervises Mary who works in the library and who lives in Epping.*
25. *John Miller works on Monday in E6A.*

Verb phrase coordination uses the two connectives *and* and *or* and coordinates complete verb phrases. In (*23*), the negative *does not* needs to be repeated in order to negate both conjuncts. As (*24*) shows, relative clause coordination requires that the relative pronoun (*who*) is repeated after the connective (*and*) in order to exclude potential ambiguity when a verb phrase and a relative clause coordination occur in the same sentence. Modifier coordination can only be realised with the conjunctive connective *and* but not with the disjunctive connective *or*, therefore the *and* can also be omitted in (*25*).

Conditionals. PENG Light supports conditional sentences for describing hypothetical situations and their consequences. A conditional sentence consists of a subordinated clause that specifies a condition and a main clause that expresses the consequence, for example:

26. *If John Miller works on Monday then John is in the office.*

Note that the subordinated clause and the main clause in (*26*) have the same internal structure as other PENG Light sentences if we would drop the conditional connective *if-then*.

Definitions. PENG Light also supports the specification of definitions: a definition is a relation of equivalence that consists of a term to be defined and an expression that is used to define that term. The following two sentences use the key phrase *is defined as* and can be used alternatively to state a definition:

27. *A mother is defined as a woman who has at least one child.*
28. *Every mother is defined as every woman who has at least one child.*

The definition in (*27*) sounds more natural in English but the indefinite determiner (*a*) is interpreted as an universal quantifier (*every*). The language processor generates a paraphrase that clarifies the interpretation of (*27*) which looks similar to (*28*).

2.3 Anaphora Resolution

In PENG Light, proper nouns, definite noun phrases and variables (but not personal pronouns) can be used as anaphoric expressions. PENG Light resolves anaphoric references by replacing the anaphoric expression with the most recent accessible noun phrase (= noun phrase antecedent) and indicates this replacement in a paraphrase. That means a noun phrase antecedent must be accessible to be referred to by an anaphoric expression. Proper nouns always denote the same object and are accessible from everywhere. An indefinite noun phrase is not accessible from **outside**, if it occurs under negation (*29*), if it occurs in a conditional sentence (*30*) or in the scope of an universal quantifier (*31*), or if it occurs under a disjunction (*32*):

29. *John does not teach a tutorial. *The tutorial ...*
30. *If John teaches a tutorial then Sue teaches a practical. *The tutorial*
31. *Every professor teaches a tutorial. *The tutorial ...*
32. *John teaches a tutorial or teaches a practical. *The tutorial ...*

However, a noun phrase antecedent in the *if*-part of a conditional sentence such as (*33*) is accessible from the *then*-part of the same sentence:

33. *If John teaches a tutorial then Sue does not teach the tutorial.*

And a noun phrase antecedent under a disjunction – as for example in (*34*) – is accessible if the anaphoric expression occurs in one of the subsequent disjuncts:

34. *John sends a letter to Mary or brings the letter to Mary.*

An anaphoric expression can be syntactically less specific than its noun phrase antecedent as the following examples (*35-37*) illustrate:

35. *The clever professor teaches COMP332. The professor ...*
36. *The professor X teaches COMP348. X works ...*
37. *The computer scientist works at Macquarie. The scientist ...*

If we interface PENG Light with an existing ontology, then we can additionally resolve anaphoric references that rely on realisation (i.e. computing the most specific class for an individual) and on classification (i.e. computing the subclass hierarchy). For example, the ontology (or the background knowledge) might specify that *lecturer* in (*38*) is the most specific class that the individual *John Miller* belongs to and that the class *academic* in (*39*) subsumes *lecturer*:

38. *John Miller teaches COMP249 on Monday. The lecturer ...*
39. *John Miller teaches COMP249 on Monday. The academic ...*

Note that if a definite noun phrase can not be resolved by the anaphora resolution algorithm that is built into the grammar of PENG Light, then this definite noun phrase is interpreted as an indefinite noun phrase and introduces a new object into the universe of discourse. The presence of a relative clause in a noun phrase antecedent does not play a role in the determination of a suitable antecedent.

2.4 Questions

PENG Light distinguishes two types of questions: *yes/no*-questions and *wh*-questions. *Yes/no*-questions such as (*40*) investigate whether a specific situation is true or false. And *wh*-questions such as (*41*) interrogate a particular aspect of a situation:

40. *Does John Miller teach a tutorial on Monday?*
41. *When does John Miller who convenes COMP249 teach a tutorial?*

Questions are derived in a systematic way from simple PENG Light sentences: in the case of *yes/no*-questions by insertion of a *do*-operator or by inversion, and in the case of *wh*-questions with the help of a suitable query word (*who, what, which, when, where, etc.*) and insertion of a *do*-operator. Note that *wh*-questions also exhibit a filler-gap dependency similar to relative clauses and that they can be processed using the same type of gapping mechanism.

3 Implementation

The grammar of PENG Light is implemented as a definite clause grammar with feature structures and difference lists that occur as arguments in the grammar rules [14]. The language processor builds a logical formula in TPTP notation [15] for a given input sentence or generates an output string for a given TPTP

formula. In order to achieve true bidirectionality, the TPTP formulas are annotated with syntactic information that supports the generation of declarative sentences in particular for question answering. PENG Light can serve as a high-level interface language to a first-order reasoner or a description logic reasoner or both, and we have experimented with all three options. In the following, we will show how sentences can be translated into TPTP notation and then into KRSS [16], the input language of the description logic reasoner RacerPro [17,18]. Note that the focus here is not on the reasoning engine but on the bidirectional translation from PENG Light into TPTP and back into PENG Light.[1]

3.1 Analysis of PENG Light Sentences

The grammar of PENG Light does not simply interpret the verb of a sentence as a relation between a subject and a number of complements and adjuncts. Instead the event or state that is described by the verb is reified and the variable of the reified relation is linked via a number of thematic relations [19] to the objects and individuals that participate in a sentence. This allows for an uniform interpretation of obligatory complements and optional adjuncts (and provides a convenient way to deal with n-ary relations in description logics [20]). Note that we use a flat notation for representing objects, individuals, and properties in order to translate efficiently between different notations. For example, the complex PENG Light sentence:

42. *John Miller who supervises Mary teaches at least two computing units on Monday.*

results in the subsequent TPTP formula where the logical forms derived from the content words are annotated (#) with syntactic information:

```
input_formula(university,axiom,
  (? [A]: ((named(A,john_miller)#[‘John’,‘Miller’] &
    (?[B]: (named(B,mary)#[‘Mary’] &
     ?[C]: (property(C,has_agent,A) &
            event(C,supervise)#[fin,third,sg,pres,no,no] &
            property(C,has_theme,B)&contemp(C,u))))#[rc]) &
      (? [D]: (timex(D,monday)#[‘Monday’] &
                (property(E,has_time,D)#[on] &
        (? [F]: ((‘$min_cardinality’(F,2) &
                  object(F,computing_unit)#[at,least,two]) &
          (? [E]: (property(E,has_agent,A) &
                    (event(E,teach)#[fin,third,sg,pres,no,no] &
                       (property(E,has_theme,F) &
                        contemp(E,u))))))))))).
```

These syntactic annotations can be used to support the generation process of sentences. For example, the annotation [on] indicates that the property

[1] Only a subset of PENG Light sentences can be handled by the description logic $\mathcal{ALCQHIR}_{\mathcal{R}} + (\mathcal{D}^{-})$ that is supported by RacerPro but the TPTP to KRSS translator will reject PENG Light sentences that are not in this subset.

`has_time` has been derived from a prepositional phrase with the preposition *on*, and the annotation `[fin,third,sg,pres,no,no]` signals that the event `supervise` has been derived from a finite verb in its third person singular. If we fed the above formula back to the grammar, **exactly** the input sentence (*42*) will be generated, and this is true for arbitrarily complex PENG Light sentences.

The lexicon of PENG Light contains apart from syntactic information partial logical forms for content words and function words. Here are two (incomplete) lexical entries for content words: the first one for a common noun and the second one for a transitive verb:

```
lex( wf:[computing,unit],... fol:X^object(X,computing_unit) ).
lex( wf:[teaches],... fol:E^X^Y^(property(E,has_agent,X) &
     event(E,teach) & property(E,has_theme,Y) & contemp(E,u)) ).
```

In the first example, the common noun *computing unit* is a compound noun that is stored as a list of tokens in the lexicon and its translation will result in an object in the logical form. The lexical entry for the transitive verb *teaches* illustrates that the logical form for this verb consists of a reified event that will connect the subject and the object of the sentence via thematic relations.

The final semantic representation of a sentence will be crucially influenced by the function words that establish its logical structure. For example, the subsequent lexical entries show that quantifiers are treated as generalised quantifiers which provide a substantial part of the scaffolding for the formal representation:

```
lex( wf:[every], ... fol:(X^Res)^(X^Scope)^(! [X]: (Res => Scope)) ).
lex( wf:[a], ... fol:(X^Res)^(X^Scope)^(? [X]: (Res & Scope)) ).
lex( wf:[exactly,two], ... fol:(X^Res)^(X^Scope)^(? [X]:
     ((‘$min_cardinality’(X,2) & Res) & Scope)) ).
```

A generalised quantifier is a relation between two sets `Res` and `Scope` (where `Res` is the restrictor and `Scope` is the scope). During processing of an input sentence, the term `X^Res` collects the logical forms that can be derived from the noun phrase, and the term `X^Scope` collects the logical forms that can be derived from the entire verb phrase of the sentence. The following top-level grammar rule `s` combines the logical forms for a noun phrase in `n3` with the logical forms for a verb phrase in `v3` and returns the resulting logical form `LF` in `s`:

```
s( mode:M, ... fol:LF, ... ) -->
   n3( mode:M, ... fol:(X^Scope)^LF, ... ),
   v3( mode:M, ... fol:(X^Scope), ... ).
```

In the case of an existentially quantified noun phrase in subject position, the variable `LF` will eventually be instantiated by a term of the form: `(? [X]: (Res & Scope))`. The following grammar rule `n3` shows that this term is issued by the determiner `det` and that the restrictor `X^Res` is processed in the subsequent noun phrase `n2` (before the anaphora resolution algorithm is triggered):

```
n3( mode:M, ... fol:(X^Scope)^LF, ... ) -->
    det( ...  fol:(X^Res)^(X^Scope)^LF, ... ),
    n2( mode:M, ... fol:(X^Res), ... ),
    { anaphora_resolution( ... ) }.
```

The scope `X^Scope` "flows" from the noun phrase `n3` into the verb phrase `v3` and then into `v2`. Note that the following grammar rule `v2` is only used for analysing sentences (`mode:ana`) but not for generating sentences (we will discuss this issue in more detail in Section 3.3):

```
v2( mode:ana, ... fol:X^Scope, ... )  -->
    v1( mode:ana, ... fol:E^X^V1, ... ),
    p2( mode:ana, ... fol:E^V1^Scope, ... ).
```

The logical forms for the verb and its complement(s) are processed in `v1` and the result `E^V1` (with the event variable `E`) is pushed together with the scope `Scope` into the grammar rules for the prepositional modifier (first into `p2` and then `p1`) where this information is finally combined with the logical form for the modifier `Mod`:

```
p1( mode:M, ... fol:E^V1^Scope, ... ) -->
    prep( ..., fol:E^Y^Mod, ...),
    n3( mode:M, ... fol:(Y^(Mod & V1))^Scope, ... ).
```

To complete the picture, we present below the top-level grammar rule that is used for processing declarative PENG Light sentences:

```
s( mode:M, fol:LF, gap:G1-G3, para:P1-P3, ant:A1-A3 ) -->
   n3( mode:M, syn:[subj,Per,Num], sort:_, fol:(X^Scope)^LF,
       gap:[]-G2, para:P1-P2, ant:A1-A2 ),
   v3( mode:M, crd:_, syn:[fin,Per,Num,_,_,_], fol:E^(X^Scope),
       gap:G1-G3, para:P2-P3, ant:A2-A3 ).
```

Apart from the logical form, this grammar rule deals with syntactic constraints (`syn`) and uses three additional feature structures that are implemented as difference lists: the first one (`gap`) deals with filler-gap dependencies, the second one (`para`) builds up a paraphrase during parsing, and the third one (`ant`) maintains the accessible noun phrase antecedents for anaphora resolution. Anaphora resolution is done during the parsing process: whenever a definite noun phrase or a proper noun have been processed, the anaphora resolution algorithm is triggered. This algorithm can query the description logic knowledge base and check for class membership and subsumption, and it dynamically updates the paraphrase (`para`) which clarifies the interpretation of a sentence.

The TPTP representation of a sentence (or an entire paragraph) can be further translated into KRSS notation; in our case, the translation of (*42*) results in the following assertions:

```
( related supervise_1 john_miller has_agent )
( instance supervise_1 supervise )
( related supervise_1 mary has_theme )
( relatated teach_2 monday has_time )
( instance sk_1 (at-least 2 computing_unit ))
( related teach_2 john_miller has_agent )
( instance teach_2 teach )
( related teach_2 sk_1 has_theme )
```

Note that `supervise_1` and `teach_2` are constants – without reification it would not be possible to represent the sentence (*42*) in description logic.

3.2 Analysis of PENG Light Questions

In PENG Light, questions are translated in a similar way as declarative sentences, and their processing uses most of the same grammar rules. Similar to relative clauses, *wh*-questions evoke filler gap-dependencies, for example:

43. [*When*]$_{filler}$ *does John Miller teach a computing unit* [ ]$_{gap}$*?*

This filler-gap dependency is handled in the grammar via a difference list (`gap`) that moves the filler term for the query word back into the position where the gap occurred. There are specific grammar rules that recognise gaps and remove filler terms from the difference list. The translation of (*43*) results in the following conjunctive TPTP query which contains the free variable `B` and the property `property(C,has_time,B)` that have both been derived from the query word *when* using information from the lexicon:

```
input_formula(university,conjunctive_query,
  ((? [A]: (named(A,john_miller)#[‘John’,‘Miller’] &
            (free_variable(B) & (property(C,has_time,B) &
    (? [D]: (object(D,computing_unit)#[computing,unit] &
    (? [C]: (property(C,has_agent,A) &
              (event(C,teach)#[inf,third,sg,pres,no,no] &
               (property(C,has_theme,D) &
                contemp(C,u)))))))))) => answer(B))).
```

The language processor first stores this TPTP query that will later serve as a template for generating answers and then translates the TPTP query into an nRQL query, the query language of the description logic reasoner RacerPro [18]:

```
(retrieve (?2) (and (?1 ?2 has_time) (and (?3 computing_unit)
(and (?1 john_miller has_agent) (and (?1 teach) (?1 ?3 has_theme))))))
```

Here `(?2)` is the head of the nRQL query and the rest is the body of the query. The head specifies the format of the query result and the body specifies the retrieval conditions.

3.3 Generation of PENG Light Sentences

Let us assume that the description logic reasoner RacerPro finds the following answer: `(((?2 1400)))` for (*43*). The language processor takes the TPTP formula that has been stored for the question and transforms it into a formula for a declarative sentence. In our case, it replaces the term for the free variable `B` by an existentially quantified expression `(? [B]: (timex(B,1400)))` and updates the syntactic annotation of the verbal event since the answer must consist of a finite verb (`fin`) and not an infinite one (`inf`). The transformed TPTP formula is then sent to the grammar and a complete sentence is generated as an answer.

Note that the annotated logical form drives this generation process. That means the logical form needs to be split up at various points into the relevant parts, in the case of a verbal modifier this splitting requires a specific grammar rule (`mode:gen`), otherwise generation would be blind for the obvious:

```
v2( mode:gen, ..., fol:E^X^(?[Y]: (Res & (Mod & V1))), ... ) -->
    v1( mode:gen, ..., fol:E^X^V1, ... ),
    p2( mode:gen, E^V1^(?[Y]: (Res & (Mod & V1))), ... ).
```

Here the variable `Res` contains the logical form for the temporal expression *2pm*, the variable `Mod` the logical form for the preposition *at* and the variable `V1` the logical form for the verb phrase *teaches a computing unit*. This will eventually results in the answer: *John Miller teaches a computing unit at 2pm.*

4 Conclusions

This paper presented the controlled natural language PENG Light and introduced a bidirectional grammar that can be used to translate PENG Light sentences and questions into syntactically annotated TPTP formulas and to generate answers to questions by transforming TPTP formulas for questions into TPTP formulas for declarative sentences. The novelty of this approach is that most of the same grammar rules can be used for the following three central tasks: analysing sentences, processing questions and generating answers to questions. The bidirectional grammar is written in a very modular way and only a small number of the grammar rules are task-specific. PENG Light can be interfaced directly with a reasoning engine and can serve as a high-level knowledge specification and query language. Note that the writing process of PENG Light sentences can be supported by a predictive text editor. Such a predictive text editor enforces the restrictions of the controlled natural language and guides the user of the controlled natural language via lookahead information. This lookahead information can be harvested directly from the grammar rules of the controlled natural language while a sentence is written. There are many potential applications that can benefit from a high-level interface language like PENG Light. We are currently investigating the usefulness of controlled natural language as an interface language to the Semantic Web and as a language for writing business rules.

References

1. Kittredge, R.I.: Sublanguages and controlled languages. In: Mitkov (ed.) The Oxford Handbook of Computational Linguistics, pp. 430–447. Oxford University Press, Oxford (2003)
2. Fuchs, N.E., Schwertel, U., Schwitter, R.: Attempto Controlled English – not just another logic specification language. In: Flener, P. (ed.) LOPSTR 1998. LNCS, vol. 1559, pp. 1–20. Springer, Heidelberg (1999)
3. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In: Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. LNCS, vol. 5224, pp. 104–124. Springer, Heidelberg (2008)

4. Clark, P., Harrison, P., Jenkins, T., Thompson, T., Wojcik, R.: Acquiring and Using World Knowledge Using a Restricted Subset of English. In: Proceedings of FLAIRS 2005, pp. 506–511 (2005)
5. Clark, P., Harrison, P., Thompson, J., Wojcik, R., Jenkins, T., Israel, D.: Reading to Learn: An Investigation into Language Understanding. In: Proceedings of AAAI 2007 Spring Symposium on Machine Reading, pp. 29–35 (2007)
6. Sowa, J.F.: Common Logic Controlled English. Draft (February 24, 2004) (2004), http://www.jfsowa.com/clce/specs.htm
7. Sowa, J.F.: Common Logic Controlled English. Draft (March 15, 2007) (2007), http://www.jfsowa.com/clce/clce07.htm
8. Schwitter, R.: English as a Formal Specification Language. In: Proceedings of DEXA 2002, Aix-en-Provence, France, September 2-6, 2002, pp. 228–232 (2002)
9. Schwitter, R., Tilbrook, M.: Meaningful Web Annotations for Humans and Machines using Controlled Natural Language. Expert Systems 25(3), 253–267 (2008)
10. Schwitter, R., Ljungberg, A., Hood, D.: ECOLE – A Look-ahead Editor for a Controlled Language. In: Proceedings of EAMT-CLAW 2003, Dublin City University, Ireland, May 15-17, pp. 141–150 (2003)
11. ASD STMG: ASD Simplified Technical English. Specification ASD-STE100, International specification for the preparation of maintenance documentation in a controlled language, Issue 4 (January 2007)
12. Thompson, C.W., Pazandak, P., Tennant, H.R.: Talk to Your Semantic Web. IEEE Internet Computing 9(6), 75–79 (2005)
13. Kamp, H., Reyle, U.: From Discourse to Logic. In: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. Kluwer, Dordrecht (1993)
14. Pereira, F.C.N., Shieber, S.M.: Prolog and Natural-Language Analysis, CSLI Lecture Notes, Number 10 (1987)
15. Sutcliffe, G., Suttner, C.B.: The TPTP Problem Library: CNF Release v1.2.1. Journal of Automated Reasoning 21(2), 177–203 (1998)
16. Patel-Schneider, P.F., Swartout, B.: Description-Logic Knowledge Representation System Specification from the KRSS Group of the ARPA Knowledge Sharing Effort, November 1 (1993)
17. Haarslev, V., Möller, R.: Racer: A Core Inference Engine for the Semantic Web. In: Proceedings of EON 2003, pp. 27–36 (2003)
18. Racer Systems: RacerPro User's Guide, Version 1.9.2, Technical report (2007), http://www.racersystems.com
19. Parsons, T.: Events in the Semantics of English: A Study in Subatomic Semantics. MIT Press, Cambridge (1994)
20. Noy, N., Rector, A.: Defining N-ary Relations on the Semantic Web, W3C Working Group Note (April 12, 2006) (2006), http://www.w3.org/TR/swbp-n-aryRelations/

Improving Metrical Grammar with Grammar Expansion

Makoto Tanji[1], Daichi Ando[2], and Hitoshi Iba[1]

[1] Graduate School of Engineering
[2] Graduate School of Frontier Science,
The University of Tokyo, Japan
{tanji,dando,iba}@iba.k.u-tokyo.ac.jp

Abstract. This paper describes Metrical PCFG Model that represents the metrical structure of music by its derivation. Because the basic grammar of Metrical PCFG model is too simple, we also propose a grammar expansion method that improves the grammar by duplicating a nonterminal symbol and its rules. At first, a simple PCFG model which that represent the metrical structure by a derivation just like parse tree in natural language processing. The grammar expansion operator duplicates a symbol and its rules in the PCFG model. Then the parameters of PCFG are estimated by EM algorithm. We conducted two experiments. The first one shows the expansion method specialized symbols and rules to adapt to the training data. Rhythmic patterns in a piece were represented by expanded symbols. And in the second experiment, we investigated how the expansion method improves performance of prediction for new pieces with large corpus.

1 Introduction

In the field of music research, it is widely recognized that music has some structures. Especially a metrical structure is important to analyze and recognize music pieces. Listener perceives the metrical structure consciously and subconsciously for almost all music pieces. In order to analyze and generate music pieces by means of computational method, a suitable model for music structure is needed. Lerdahl and Jackendoff have presented a framework consisting of four kinds of hierarchical structure: grouping structure, metrical structure, time-span reduction and prolongational reduction in their *Generative Theory of Tonal Music* [8] (GTTM). The metrical structure consists of several levels of beat patterns as pointed out in GTTM. From the viewpoint of its hierarchy, music and language have similar features. Therefore, musical researches using methods of natural language processing have a long history.

Most of past works have used Markov Models, Hidden Markov Models, N-gram Models and so on. From a perspective of theory of generative grammar, there is the more powerful model called CFG (Context-Free Grammar). In recent times, some studies using stochastic and linguistic model for music information processing have been researched. Gilbert and Conklin [3] used a PCFG (Probabilistic CFG) for melodic reduction. And Bod [2] analyzed phrasing structure of both music and language in a unified model. Yamamoto, et al. [13] also used PCFG to detect music rhythm and tempo.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 180–191, 2008.

In this paper, we propose a metrical model to analyze the metrical structure based on PCFG. Because PCFG does not capture contexts, a basic metrical grammar is too simple to represent any rhythmic patterns. Thus, we also present a grammar expansion method for the metrical grammar that expands symbols and rules to adapt for training data.

We first give a brief description of the metrical structure in Section 2, and then Section 3 describes our metrical PCFG model. Section 4 describes grammar expansion method. Section 5 reports on evaluation experiments and results. Finally conclusion stemming from this investigation is summarized in Section 6.

2 Metrical Structure

Human listener perceives music as the entire rhythmic pattern rather than each individual musical note. The structure of several levels of beats is called *Metrical Structure*. Listener can clap hands at various length of beats which may be quarter length or length of a musical bar. Fig.1 shows an example by Metrical Grid [12]. There are four levels of beats in the figure, length of eighth note, quarter note, half note and whole note. A beat which has many dots indicates beat of "higher" level and includes two or three of relative "lower" beats in it. Therefore, the metrical structure is represented by hierarchical form [8] [12]. The right part of the figure is a tree representation for the same note sequence.

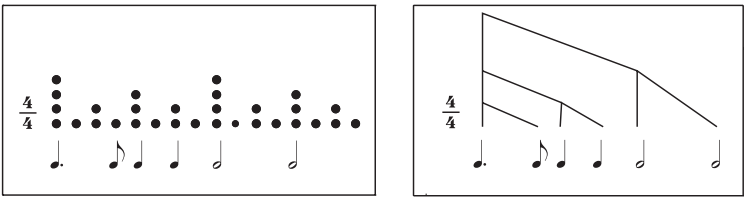


Fig. 1. Representation of Metrical Structure

Human listener usually can recognize an original (intended by player and composer) structure. However, there are many possible structures for one note sequence. As we can see in Fig.2, there are several possible structures for one note sequence. There are different beat types, 4/4, 3/4, 6/8 .... Furthermore, for the same beat type, different understandings can be considered by starting from the different position.

The goal of this research is to construct computational model that estimates the same structure as human listener and emulates music knowledge of it. Furthermore, more large structure like repeats and parallelism are important to understand score in a broader perspective. The suitable computational model provides us a tool for preprocessing of the analysis of music pieces deeply. And the metrical structure estimation has many potential applications such as harmony estimation, segmentation genre classification.

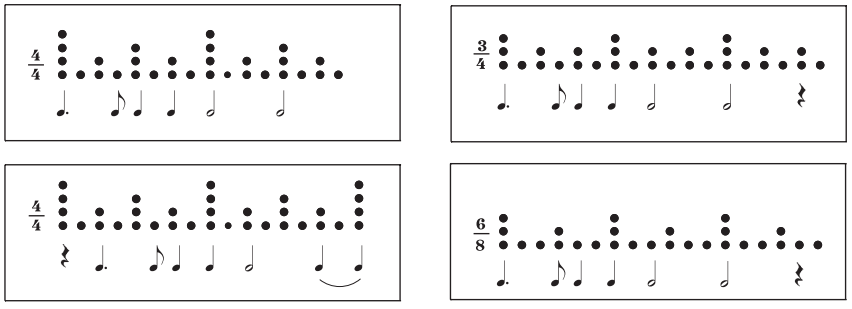


Fig. 2. Metrical Structures for the same note sequence

3 Metrical Grammar for Model

First of all, we assumed music model as illustrated in Fig. 3. A musical piece is generated from Music Model $\mathbf{G}$ with the conditional probability $P(\mathbf{T}|\mathbf{G})$ (Generation Process). The generated piece contains both the note sequence and the musical structure (includes the metrical structure). However, we can observe only note sequence or score that does not contains full structure (Note Observation). The performance data $\mathbf{W}$ by human player is also represented stochastic process with the conditional probability $P(\mathbf{W}|\mathbf{S})$.

Then, we assume Music Model $\mathbf{G}$ to be PCFG (Probabilistic Context-Free Grammar). From the view point of the above model, the structure estimation problem is regarded as an inverse problem that takes a note sequence $\mathbf{S}$ and outputs the original structure $\mathbf{T}$. Analogously, the score estimation problem is a problem that takes performance note sequence $\mathbf{W}$ and outputs the original score (the structure $\mathbf{T}$ and the note sequence $\mathbf{S}$). By modeling these probabilities $P(\mathbf{T}|\mathbf{G}), P(\mathbf{W}|\mathbf{S})$, we can formulate these problems mathematically.

According to the above stochastic model, we propose a model named *Metrical PCFG Model* and a method for estimation of note sequence and metrical structure using the model. Our approach is based on the assumption that note sequence and metrical structure are generated from Metrical PCFG Model with probability $P_G(\mathbf{T})$. And only note sequence $\mathbf{S}$ is observed ($P(\mathbf{S}|\mathbf{T}) = 1$). Therefore, the problems of metrical structure estimation is given as the reverse problem that is to maximize conditional probability from observed music $\mathbf{S}$.

$$\hat{\mathbf{T}} = \underset{\mathbf{T}}{\operatorname{argmax}} P(\mathbf{T}|\mathbf{S}) = \underset{\mathbf{T}}{\operatorname{argmax}} \frac{P(\mathbf{S},\mathbf{T})}{P(\mathbf{S})}$$
$$= \underset{\mathbf{T}}{\operatorname{argmax}} P(\mathbf{S}|\mathbf{T})P_G(\mathbf{T}) = \underset{\mathbf{T}}{\operatorname{argmax}} P_G(\mathbf{T})$$

3.1 PCFG (Probabilistic Context-Free Grammar)

PCFG (Probabilistic Context-Free Grammar) is a model developed from CFG (Context-Free Grammar) to be a stochastic model by attaching probabilities to production rules. A PCFG G is defined as follows.

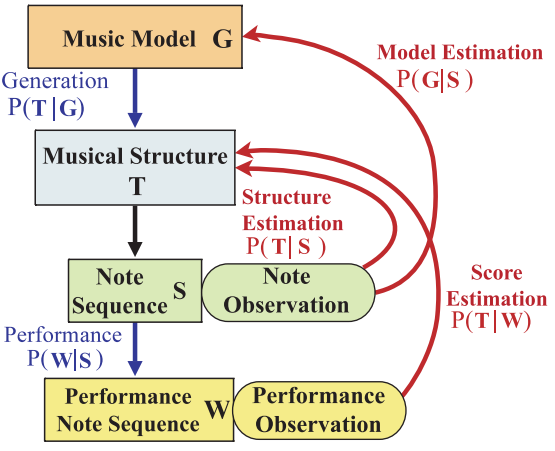


Fig. 3. Overview of the method

- $G = \{V_T, V_T, P, S\}$
- V_N : a finite set of nonterminal symbols
- V_T : a finite set of terminal symbols
- P : a finite set of production rules $< A \rightarrow \alpha, p >$ ($A \in V_N, \alpha \in (V_N \cup V_T)^*$, p : probability)
- S : start symbol ($S \in V_N$)

A process that starts from symbol S and successively applies production rules and then obtains sequence of terminal symbols is called *derivation*. And obtained sequence of terminal symbols is called *string*. We regard the derivation and the string as the metrical structure **T** and the music notes sequence **S** in this paper. A derivation **T** of a sequence **T** is represented as tree structure and a probability is given by

$$P_G(\mathbf{T}) = p(t_1) \cdots p(t_N) = \prod_{i=1}^{N} p(t_i)$$

where t_i is the i-th production rule applied in the derivation of **T**.

Grammar **G** is called an ambiguous grammar if there are more than one derivation **T** for one string **S**. PCFG allows us to distinguish derivations by comparing probabilities and determine the maximum likelihood derivation as the most probable one.

3.2 Metrical PCFG Model

We construct a simple PCFG grammar for metrical model as shown in Table 1. The number of parameters of production rules is 148. This grammar derives a note sequence, applying production rules. It can generate melodies that starts with auftact (upbeat) and ends with incomplete measure by applying rules marked by *1 and *2. And the metrical structure for derived note sequence corresponds to derivation tree just like parse tree in natural language processing.

Table 1. Grammar of Metrical PCFG

Nonterminal Symbols	**Terminal Symbols**
ST, Beat$x$, Ny ($x \in \mathbf{A}, y \in \mathbf{B}$)	note:x[p], rest:x ($x \in \mathbf{B}$, $p \in \mathbf{P}$)
Production Rules	
ST $\rightarrow$ Nx Beat$x$ ($x \in \mathbf{A}$)	
ST $\rightarrow$ Nx Beat$y$ ($x \in \mathbf{B}, y \in \mathbf{A}, x < y$)	$*1$
ST $\rightarrow$ Nx Nx ($x \in \mathbf{A}$)	
Beat$x$ $\rightarrow$ Nx Beat$x$ ($x \in \mathbf{A}$)	
Beat$x$ $\rightarrow$ Nx Nx ($x \in \mathbf{A}$)	
Beat$x$ $\rightarrow$ Nx Ny ($x \in \mathbf{A}, y \in \mathbf{B}, y < x$)	$*2$
Nx $\rightarrow$ Ns Nt ($x, s, t \in \mathbf{B}$, s + t = x)	
Nx $\rightarrow$ note:x[p] ($x \in \mathbf{B}, p \in \mathbf{P}$)	
Nx $\rightarrow$ rest:x ($x \in \mathbf{B}$)	
$\mathbf{A}$ = {1/1 3/4}, $\mathbf{P} = \{1 \ldots 7\}$	
$\mathbf{B}$ = {1/1, 1/2, 1/4, 1/8, 1/16, 1/32, 3/4, 3/8, 3/16, 3/32}	

The procedure of a derivation is the following.

1. Produces "Nx Beat$x$" from start symbol ST. The symbol x corresponds to time signature (e.g. ST $\rightarrow$ N1/1 Beat1/1).
2. Produces "Nx ... Nx" appling the rule "Beat$x$ $\rightarrow$ Nx Beat$x$" until the rule "Beat$x$ $\rightarrow$ Nx Nx".
3. Divides "Nx" into smaller length "Ns" "Nt" recursively, or becames terminal symbol "note:x[p]". (x and p indicate length and pitch.)
4. Ends if all symbols are terminal.

For instance, Fig. 4 shows a part of sample derivation tree. In this case, $N1/1$ means one musical measure.

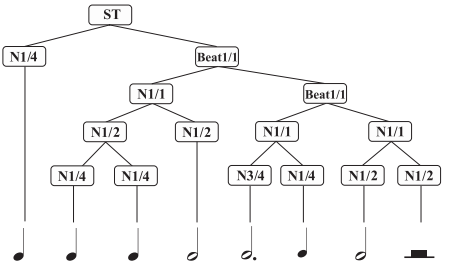


Fig. 4. Sample tree of Metrical PCFG

The grammar of Metrical PCFG Model is ambiguous because it has more than one derivations for one note sequence. For example, two different derivations are shown in Fig.5. In hearing the note sequence, listener will likely perceive the first metrical

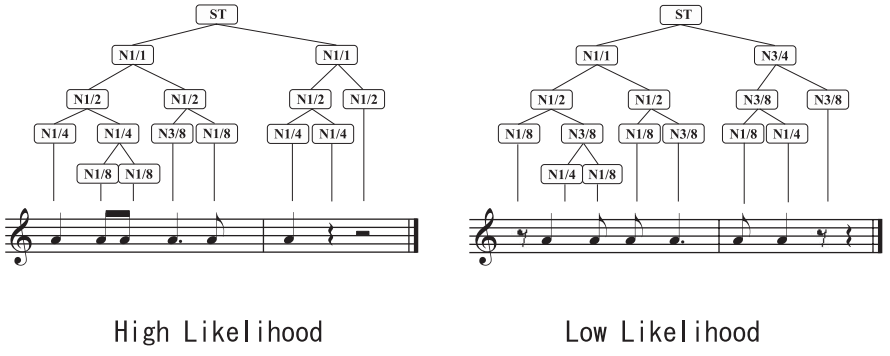


Fig. 5. High and Low likelihood Structures

structure rather than the second one. This difference can be explained in terms of probability. The production rule "N1/2 → N1/4 N1/4" is considered to be common event. In contrast, the production rule "N1/2 → N1/8 N3/8" is not so common event (let us assume this rule is not common). Therefore, the probability of the second one will be low compared with the first one. Thus, we can choose a probable structure by considering probability of derivation.

3.3 Maximum Likelihood Estimation and Parameter Estimation

Metrical Structure Estimation. The maximum likelihood derivation $\hat{\mathbf{T}} = \operatorname{argmax}_T P_G(\mathbf{T})$ is calculated by Viterbi algorithm using CYK algorithm in polynomial time [7].

Score Estimation. This is the problem that estimates the original metrical structure and quantized note sequence from performed music. It also can calculated by CYK-like algorithm. See more detail in our previous paper[11].

Grammar Estimation. And *Inside-Outside algorithm* is known to estimate parameters efficiently as a variant of the EM (Expectation Maximization) algorithm [6][4]. According to inside-outside algorithm, estimated appearance count of a rule $count(A \to \alpha; G)$ in sentence S is estimated with a current grammar G. And estimated count of a symbol A is calculated as $count(A) = \sum_\alpha count(A \to \alpha; S)$. The parameter re-estimation procedure is written as

$$\bar{P}(A \to \alpha) = \frac{count(A \to \alpha)}{\sum_\beta count(A \to \beta)}$$

An efficient algorithm for partially bracketed data has been proposed [9]. It allow us to save computational cost because brackets reduce ambiguity of derivations. We use these algorithms to estimate the maximum likelihood derivation and parameters of the grammar.

4 Grammar Improving

The metrical grammar we described in previous section can generate various rhythms of melodies and distinguish derivations by its probabilities. However, real music have

some context between musical blocks. It means there are rhythm patterns, idioms. And a particular pattern have relatively high probability. Our model is very simple and compact to distinguish various patterns of music. Therefore, we describe a grammar expansion approach to capture musical patterns and context. Some different approaches to refine stochastic grammar from a compact grammar were proposed [5][1].

We perform a symbol expansion approach that duplicates a nonterminal symbol and related production rules ($X\ to\ X,\ X'$). After the expansion, EM (Expectation Maximization) algorithm is applied to adapt parameters to training data. As a result, two symbols are expected to represent different music pattern if training data have biased frequency of rhythms. This algorithm simply increases the likelihood because duplication does not influence the likelihood and the EM algorithm monotonically increase the likelihood. The pseudo code of the grammar expansion algorithm is illustrated below.

Algorithm 4.1: GRAMMAR EXPANSION(G, C)

$G := grammar$
$\theta\ := parameters$
$C := Corpus$

$\#\ initialization$
$G \leftarrow basic\ grammar$
$\theta \leftarrow initial\ value$
for $i \leftarrow 1$ **to** $N\ \ \#\ expand\ N\ times$
do $\begin{cases} \textbf{for}\ i \leftarrow 1\ \textbf{to}\ M\ \ \#\ parameter\ estimation\ by\ EM\ algorithm \\ \quad \textbf{do}\ \theta \leftarrow EMStep(G, \theta, C) \\ \#\ grammar\ expansion \\ X \leftarrow selectSymbol(G, \theta) \\ (G, \theta) = expand(G, \theta, X) \end{cases}$
return (G, θ)

4.1 Expansion Operator

In this section, we describe details of the Expansion operator. The Expansion operator takes one nonterminal symbol *X* and duplicates it (*X'*), then all the production rules related to *X* are also duplicated. An example of expansion *X* is shown below.

$$\begin{bmatrix} A \rightarrow B\ X \\ A \rightarrow X\ X \\ X \rightarrow A\ B \end{bmatrix} \Longrightarrow \begin{bmatrix} A \rightarrow B\ X & A \rightarrow X\ X \\ A \rightarrow B\ X' & A \rightarrow X\ X' \\ X \rightarrow A\ B & A \rightarrow X'\ X \\ X' \rightarrow AB & A \rightarrow X'\ X' \end{bmatrix}$$

The probability of each duplicated rule is assigned as

$$p_{new}(A \rightarrow \alpha\ X\ \beta) = p_{new}(A \rightarrow \alpha\ X'\ \beta) = \frac{p_{old}(A \rightarrow \alpha\ X\ \beta)}{C(A \rightarrow \alpha\ X\ \beta)}$$

$$p_{new}(X' \rightarrow \alpha) = p_{old}(X \rightarrow \alpha) + \epsilon$$

where $C(A \rightarrow \alpha\ X\ \beta)$ is the number of rules after the duplication. And ϵ is random noise to alter symmetry of duplicated symbols and distinguish its probabilities.

The selection of a nonterminal symbol to be expanded from all nonterminal symbols is determined by the following selection procedures.

Random Selection. The selection probabilities of all original nonterminal symbol are the same value $P_{random}(X) = \frac{1}{N}$. Duplicated symbols are counted as one because the more a symbol is expanded, the more it increases selection probability. Therefore the number of nonterminal symbol N does not change.

Proportionate Selection. The selection probability of a symbol is proportionate to its estimated appearance count. $P_{proportion}(X) = \frac{count(X)}{\sum_Y count(Y)}$. The duplicated count $count(X)$ and $count(X')$ are also summed up for the same reason.

5 Experimental Result

5.1 Grammar Inducion for a Single Piece

At first, we applied our grammar expansion method to single piece. The target piece was chosen to an excerpt of Bash's Menuet BWV1009, classical and simple melody Fig.6. It should be noted that the piece is constructed a few rhythmic note patterns. By considering only rhythmic element, there are only three rhythmic patterns [1/4 1/8 1/8 1/8 1/8], [1/4 1/4 1/4] and [3/4]. Therefore the specialization for the three patterns is hoped to increase the likelihood.

In other words, the information of the piece can be compressed well. From the feature, expanded symbols are expected to specialize these patterns by the grammar expansion method.

Seven types of expansion were applied, expansion of [no expansion], [N1/4], [N1/4, N1/2], [N1/4, N1/2, N3/4], [N1/4, N1/2, N3/4, Beat3/4], [N1/4, N1/2, N3/4, N3/4, Beat3/4], [N1/4, N1/2, N3/4, N3/4, Beat3/4, Beat3/4]. We expanded symbols at the 10th time of EM repetition. Fig.7 shows the log likelihood curves for each expansion. From the figure, we can observe the likelihood increases with the number of EM steps. The [no expansion] converged after a few EM steps. And the expansion [N1/4, N1/2, N3/4, N3/4, Beat3/4, Beat3/4] archived the best log likelihood value. However, more complex types of expansion had not improved the likelihood. Fig.6 illustrates a part of the maximum likelihood derivation on the grammar which expanded [N1/4, N1/2, N3/4, N3/4, Beat3/4, Beat3/4]. We assigned name 'A', 'B' and 'C' to 'N3/4', 'N3/4*' and 'N3/4**' to be easy to understanding in fig.6. Because each rule of symbol [N3/4, N3/4*, N3/4**] has different probability and specializes to each rhythmic pattern, 'A', 'B' and 'C' appeared at reasonable places in maximum likelihood derivation. The behavior imply that the grammar expansion method makes the grammar to adapt to the training data if appropriate symbols are selected.

5.2 Grammar Expansion for Large Corpus

An objective measure of performance of our grammar expansion method is how it predict the metrical structure for new pieces (not appear in training data). Therefore we tested the effectiveness of our method on large corpus.

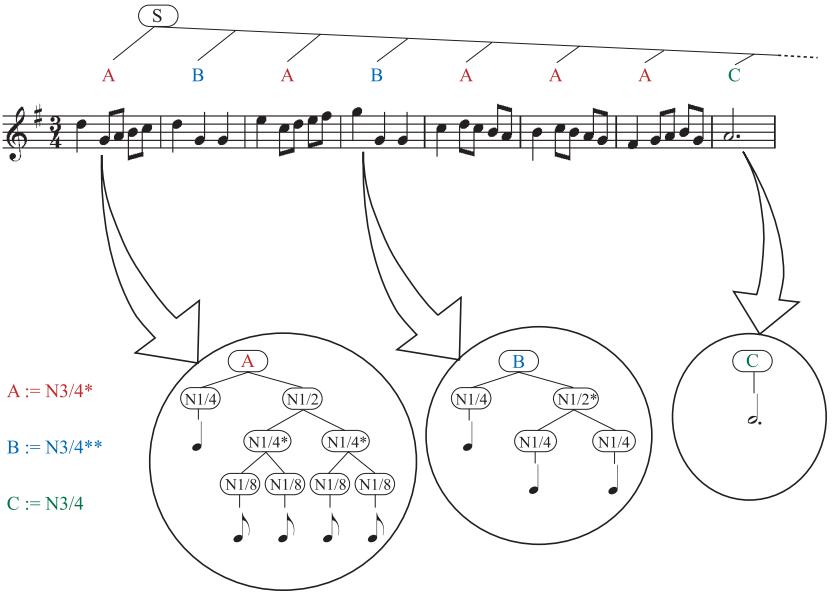


Fig. 6. Assigned symbols for a experpt of Menuet by J.S.Bach

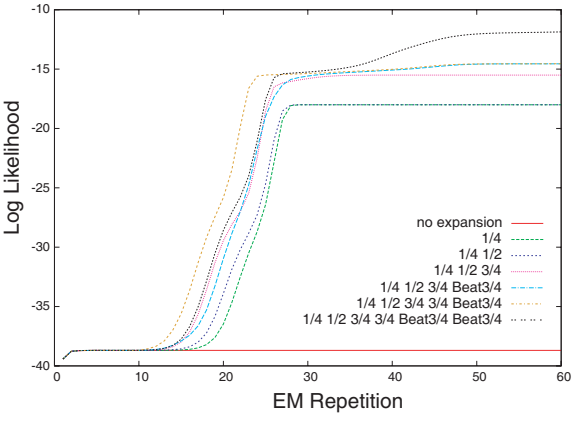


Fig. 7. The log likelihood graph in 6 types of expansions

Corpus. The Essen Folksong Collection [10] was used for the music corpus in the experiment. The Essen Folksong Collection contains large amount of monophony folksongs. Each song includes a key signature, a beat signature, bar boundaries, phrase boundaries and other optional text information. To simplify the problem, we removed songs which have irregular meter and selected songs of 4/4, 3/4, and 6/8 beat. Selected songs were converted to following format.

$S_{parenthesis}$={(1/4[1] 1/4[2] 1/4[3] 1/8[4] 1/8[5]) (1/2[6] 1/2[5]) . . . }

For each note, the first fractional number indicates note length, and [x] is pitch number at scale degree. The parenthesis '(', ')' means bar boundary.

Consequently, we got about 3300 songs, then divided it into 90% and 10% as the training data and the test data. Each ratio of the beat type of 3/4, 4/4, 6/8 was 30%, 40%, 30% respectively. While the training data contains bar boundaries, we removed bar boundaries on test data.

Grammar Improving. The basic metrical grammar was expanded and trained by training data according to algorithm described in Section 4. The expansion times N and the number of EM step M were 20 and 10 respectively. After each expansion and training, the metrical structure of test data was obtained from maximum likelihood derivation by CYK algorithm, and compared with correct bar boundaries in the corpus. After each expansion and training, the model was tested with the test data. The metrical structure of test data was estimated from the maximum likelihood derivation by CYK algorithm, and compared with correct bar boundaries in the corpus.

TP (True Positive), FP (False Positive), FN (False Negative) and TN (True Negative) between the estimated structure and the corpus were calculated. And F-Score was used to evaluate how well positions of bar boundaries were recognized. The *recall* and *precision*, and *F-score* are defined in following.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$
$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$
$$F - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Result. The results of the experiment are summarized in Fig.8. There are two selection method 'proportionate selection' and 'random selection'. The left graph indicates the log likelihood curves for the training data. The right graph is the F-score for the test data. We can see that the log likelihood curves monotonically increase as the number of expansion rises. Then, both curves of the F-score of random selection and proportion selection show that grammar expansion increases accuracy of prediction for the test data. If there was no expansion, the parameters of the grammar would not change. These results imply that expanded grammar found some common rhythmic patterns which generally exist in the corpus. If expansion symbol adapted only for the training data, it should not increase. The average increase of proportionate selection and random selection is about 7% and 6% respectively.

Table 5.2 gives values of TP (true positive), FP (false positive), FN (false negative) and TN (true positive) for the best model after the 17th expansion (Proportional Selection). The Cohen's Kappa statistic was calculated as an another measure of performance (CK=0.838).

When one compares the proportionate selection and the random selection, one sees that the proportionate selection is better likelihood and F-score at the end of expansion. However, there are little differences between two selection methods at least the first phase of expansion. This may be due to that the random selection does not work well

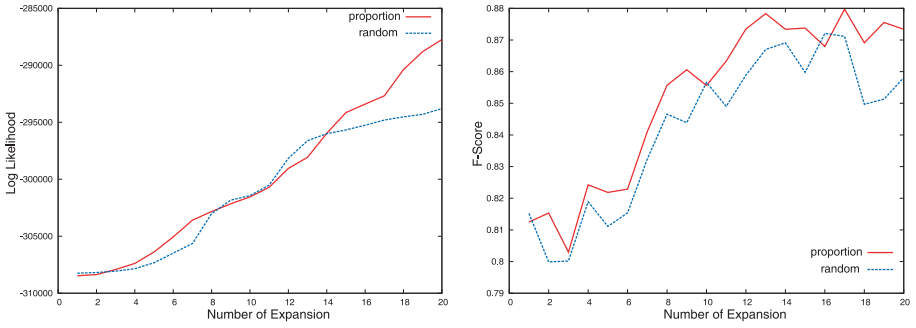


Fig. 8. The log likelihood data on the training data (left), and the F-score of maximum likelihood derivation on the test data vs. the number of expansion

Table 2. Result of the best model

		Estimation	
		Boundary on	off
Correct	Boundary on	11568 (tp)	1522 (fn)
	off	1616 (fp)	35289 (tn)

when there are many duplicated symbols because increase of duplicated symbols may produce symbol that rarely appears. The proportionate selection increased the likelihood at the end phase of expansions, while the random selection poorly increase at the end. A possible interpretation of the result is that the proportionate selection selects promising symbols because of the probability of the estimated appearance count.

In the experiment, we observed that F-score increases at first stage with expansion, reaches plateau region, and then sometimes decrease. The most likely cause is overfitting for the training data. The fact implies the necessity for a measurement of when expansion should stop like cross-validation method. More detailed work is necessary to understand the behavior.

6 Conclusion

We proposed the metrical PCFG model and method of grammar expansion to improve basic PCFG grammar by expansions of nonterminal symbols. The grammar expansion and EM algorithm allow basic grammar to adapt to training data by specializing probabilities for musical patterns. As a result of adaptation, duplicated symbols are assigned to each music pattern from scratch. We observed that our grammar expansion method increased accuracy of prediction for new music pieces as well as training data. The basic grammar uses only the dividing rule of rhythm, no other musical knowledge is given. Further developments in the other grammar refinement operations, for example rule removal and concatenation of symbols, are possible to explore suitable grammars.

References

1. Bockhorst, J., Craven, M.: Refining the structure of a stochastic context-free grammar. In: IJCAI, pp. 1315–1322 (2001)
2. Bod, R.: Probabilistic grammars for music. In: Belgian-Dutch Conference on Artificial Intelligence (2001)
3. Gilbert, E., Conklin, D.: A probabilistic context-free grammar for melodic reduction. In: International Workshop on Artificial Intelligence and Music (2007)
4. Kita, K.: Probabilistic Language Model (Japanese). University of Tokyo Press (1999)
5. Kurihara, K., Kameya, Y., Sato, T.: Efficient grammar induction algorithm with parse forests from real corpora. Transactions of the Japanese Society for Artificial Intelligence 19, 360–367 (2004)
6. Lafferty, J.D.: A derivation of the inside-outside algorithm from the em algorithm. Technical report, IBM Research Report (1993)
7. Lari, K., Young, S.J.: The estimation of stochastic context-free grammars using the inside-outside algorithm. Computer speech & language 4, 237–257 (1990)
8. Lerdahl, F., Jackendoff, R.: A Generative Theory of Tonal Music. The MIT Press, Cambridge (1983)
9. Pereira, F., Schebes, Y.: Inside-outside reestimation from partially bracketed corpora. In: The 30th Annual Meeting of the Association for Computational Linguistics, pp. 128–135 (1992)
10. Schaffrath, H.: The Essen Folksong Collection in the Humdrum Kern Format. Center for Computer Assisted Research in the Humanities, Menlo Park (1995)
11. Tanji, M., Ando, D., Iba, H.: Musical rhythm parsing using mixture probabilistic context-free grammar. In: 10th International Conference on Music Perception and Cognition, ICMPC10 (2008)
12. Temperley, D.: An evaluation system for metrical models. Computer Music Journal 28(3), 28–44 (2004)
13. Yamamoto, R., Takeda, H., Sako, S., Sagayama, S.: Estimation of rhythm and tempo of midi performance using stochastic context-free grammar. In: ASJ Autumn Meeting, pp. 571–572 (2006)

FrameNet-Based Fact-Seeking Answer Processing: A Study of Semantic Alignment Techniques and Lexical Coverage

Bahadorreza Ofoghi, John Yearwood, and Liping Ma

Centre for Informatics and Applied Optimization
University of Ballarat
PO Box 663, Ballarat VIC 3353, Australia
{b.ofoghi,j.yearwood,l.ma}@ballarat.edu.au

Abstract. In this paper, we consider two aspects which affect the performance of factoid FrameNet-based Question Answering (QA): i) the frame semantic-based answer processing technique based on frame semantic alignment between questions and passages to identify answer candidates and score them, and ii) the lexical coverage of FrameNet over the predicates which represent the main actions in question and passage events. These are studied using a frame semantic-based QA run over the TREC 2004 and TREC 2006 factoid question sets.

Keywords: Fact-Seeking Question Answering, FrameNet, Lexical Coverage.

1 Introduction

Event-based association information of predicates can be encapsulated in *semantic frames* with slots representing participant *roles* and frames containing the whole *scenario* of an event. In a similar way, the FrameNet [1] project has constructed a network of inter-related semantic frames which encapsulates Frame Semantics (FS) [2] [3]. The two main entities in FrameNet are *frames* and *Frame Elements* (FEs) where a frame describes an event or scenario in which different roles or FEs participate. Each frame covers a list of predicates that share the same semantic background. Figure 1 shows an example frame and some of its FEs.

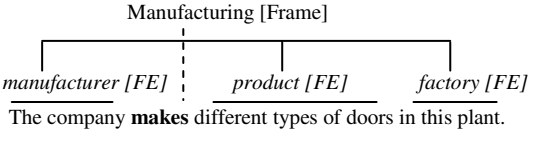


Fig. 1. An example FrameNet frame "Manufacturing" evoked by the predicate "make"

The various contributions of this linguistic resource to the domain of Question Answering (QA) have been *partially* studied along the three different dimensions shown in Figure 2. The first work in [4] and following studies such as [5], [6], [7], [8], [9],

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 192–201, 2008.

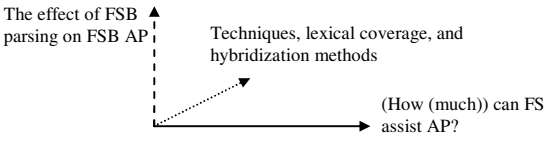


Fig. 2. The 3-D space of studying the frame semantic-based answer processing

and [10] have been conducted through measuring if and how the encapsulated FS in FrameNet can assist the task of Answer Processing (AP). Another recent study in [11] is more specifically committed to this direction by formulating the usage of frame semantic role labeling via bipartite graph optimization and matching for AP. One of the recent efforts in the second direction can be found in [12] which studies the impact of different levels of FS annotation of texts on fact-seeking (factoid) AP performance. Their work, however, does not consider the other aspects shown in Figure 2. In this paper, we study two questions to shed more light on the different aspects of using FrameNet in factoid AP:

- How is the effectiveness of a FSB AP module affected by different techniques of frame semantic alignment and what is the most effective semantic alignment method in identifying and scoring answer candidates?
- What is the effect of the lexical coverage of FrameNet over different predicates on the FSB AP task and which part-of-speech (POS) predicate(s) require more attention through enhancing factoid QA?

2 Setting of Experiments

2.1 Experimental QA System

We have developed an experimental QA system the pipelined architecture of which is shown in Figure 3.

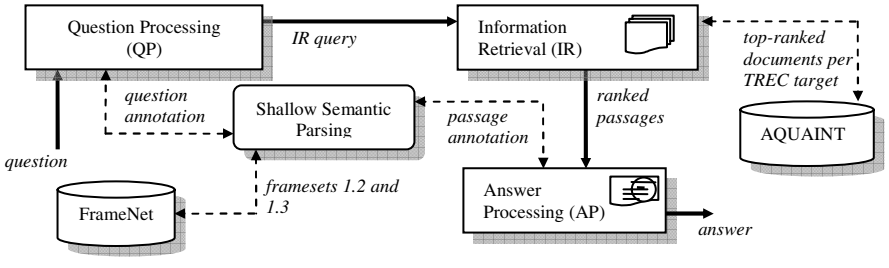


Fig. 3. The pipelined architecture of the experimental QA system

In the Question Processing (QP) module, stop-words are removed, the keywords are stemmed using the Porter stemmer, and the TREC target references are resolved by adding TREC targets to the query in cases where there is no explicit reference to

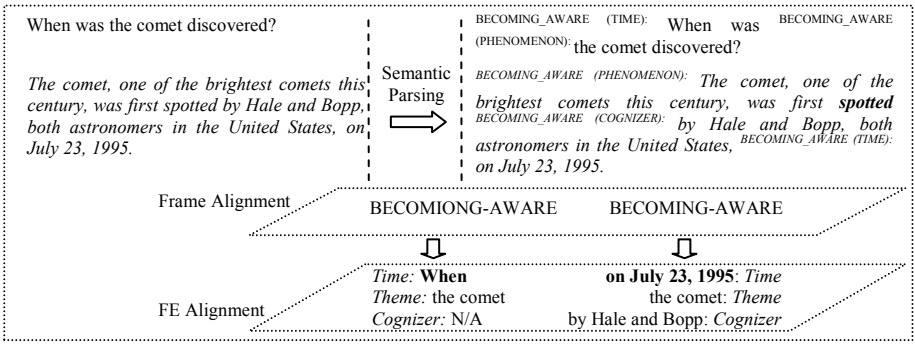


Fig. 4. AP using frame semantic alignment in the FSB model

the target concept. In the Information Retrieval (IR) module, passage-sized texts are retrieved from TREC-reported documents per target using a modified version of the MultiText passage retrieval algorithm [13, 14]. The AP module uses a *Frame Semantic-Based* (FSB) model of AP. In this model, questions and answer-bearing passages are annotated with FrameNet frames and FEs (using SHALMANESER [15]). The AP task includes frame and FE alignment to instantiate the vacant FE in questions and score candidate answers. Figure 4 illustrates an FSB AP example.

2.2 Data

Subsets of the TREC 2004 and 2006 factoid question sets are used for evaluations. Table 1 summarizes the filtering procedure of the two datasets. The AQUAINT collection is the answer resource for both datasets. The questions and passages in the TREC 2004 dataset are automatically annotated with the FrameNet 1.2 dataset and manually corrected with FrameNet 1.3 [*reference to be given after blind reviewing*]. In manual correction, frames are still restricted to FrameNet framesets. No manual correction is performed on the TREC 2006 dataset (because of time limitations).

Table 1. Dataset filtering schema – FSB error analysis is based on frame evocation and frame structure of questions and passages

Dataset	Total #questions	No answer @10 passages	N/A after FSB error analysis	Remaining set
TREC 04 factoid question set	230	87	68	75
TREC 06 factoid question set	403	227	*not performed*	176

3 FrameNet-Based Answer Processing Techniques

3.1 Methods

We have developed five different techniques for AP using FrameNet. The first and second methods are believed to have been used in previous works such as [4] and [10], while the other techniques are our new approaches to AP using FrameNet.

- *Frame alignment with complete FE alignment and no frame scoring (CFFE)*: this method finds any matching frame in passages and aligns the FEs in order to instantiate the vacant FE of the question with the value of the corresponding FE of the frame in passages. The complete FE alignment refers to the strategy where a frame is a match when the instance values of all FEs (except for the vacant FE) are matches according to a partial string matching procedure. The answers using this method are scored only on the basis of passage scores.
- *Frame alignment with specific FE alignment and no frame scoring (FSFE-NFS)*: in this method, frames are matches only if they have the same name. An answer candidate is the instance value of the vacant FE in the passages with no restriction over the other FEs of the frame.
- *Frame alignment with specific FE alignment and frames scored (FSFE-FS)*: as there will be a number of match frames in the specific alignment strategy, frame scoring on the basis of any possible pieces of evidence is crucial. Accordingly, this method scores match frames according to two aspects:
  - The instance value in the vacant FE – if there is an instance value in the FE of the passage frame corresponding to the vacant FE in the question frame, then we add 1.0 to the initial score (passage score) and add 0.0 otherwise.
  - Query term frequency – the score of a frame is added up with the raw term frequencies of each query term in the frame-bearing sentence.
- *FE alignment with no FE scoring (FE-NFES)*: this method is a big step towards making the AP strategy more shallow as there is no frame matching performed prior to FE matching. The FEs are match FEs only if they share the same name. This method, with respect to the characteristics of FrameNet where there are FEs with the same name in different frames (especially non-core common FEs like the FE "Time"), goes beyond the boundaries of FrameNet frames and the semantic information they encapsulate. However, it is still bounded to the semantic roles (FEs) assigned to the text which keep the method ahead of simple information extraction-based methods. The candidate answers (the FE instance values) are assigned passage scores.
- *FE alignment with FEs scored (FE-FES)*: this is similar to FE-NFES with the difference in the scoring scheme. The FEs are scored based on two issues:
  - Score of their parent frame – the parent frames are scored with their passage scores and the accumulated query term frequencies. The frame scores are the initial scores for the FEs.
  - Instance values – each FE score is added up with 1.0 if its instance value is not null or an empty string.

In all of the scoring and ranking methods, answer redundancy is considered as a parameter for boosting the score of the multiple-occurrence answers. The score of all answers is multiplied by their probability in answer lists and the lists are re-sorted with new scores. The probability of each answer is the ratio of its raw frequency divided by the total number of answers in the list.

3.2 Conceptual Analysis

The different FrameNet-based AP methods can be compared with respect to two aspects: i) the chance of finding matching passage elements with question elements, and

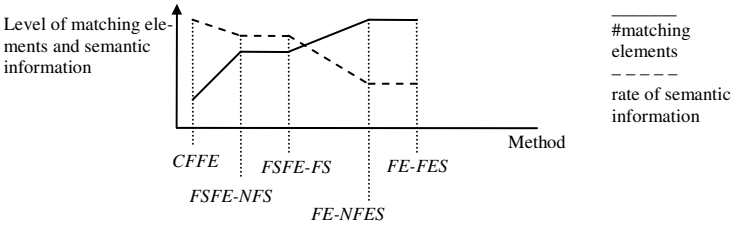


Fig. 5. The level of matching elements and semantic information in the different FrameNet-based AP methods

ii) the level of semantics that they take into consideration. Figure 5 illustrates these two aspects of the different FrameNet-based AP methods.

The CFFE method has the lowest chance of finding matching elements (frames and FEs) in the presence of different types of textual mismatches between FE instances, although it considers the maximum amount of semantic background when aligning questions and passages. This is contrary to the FE-based methods (FE-NFES and FE-FES). A balanced approach is considered in the FSFE-NFS and FSEF-FS methods. The actual AP performance of the different methods is reported in the next section.

3.3 Experimental Results and Discussion

We have run the so different methods for the FSB model of AP on the experimental datasets and tested the statistical significance of their performance differences. Table 2 summarizes the results obtained with the *strict*[1] TREC evaluation of *single* answers per question.

According to the results in Table 2, in the 2004 runs, the FSFE-FS is outperforming the other techniques with a Mean Reciprocal Rank (MRR) of 0.627. The first observation is the dramatic rise when relaxing frame alignment to the extent that not all FEs are required to be partial matches. The change from 0.293 to 0.587 clearly makes the point that tight FE matching over all of the FEs with instance values (excluding the null string FEs) in the frames is not a reasonable requirement for frame matching in the presence of different types of textual noise and/or string mismatches.

Table 2. QA runs with different FSB methods – CFFE is the baseline for significance analysis

FSB method	MRR	
	TREC 2004	TREC 2006
CFFE	0.293	0.006
FSFE-NFS	0.587 *p<0.001*	0.011 *p=0.281*
FSFE-FS	**0.627** *p<0.001*	0.011 *p=0.281*
FE-NFES	0.400 *p=0.085*	0.011 *p=0.281*
FE-FES	0.413 *p=0.062*	0.011 *p=0.281*

[1] Answers in the strict evaluation are required to have been extracted from a list of reported related documents.

Table 3. QA runs on 75 TREC 20004 factoid questions – combined settings are constructed by manual judgments of the two AP models – *p-values* are calculated with respect to the Best-TREC system – (*method**-first refers to the combination process where the second method is activated only if the *method** fails in retreiving a correct answer)

AP system	MRR	
Best-TREC	0.867	
Best-FSB (=FSFE-FS)	0.627	
Combined (Best-TREC-first)	**0.947**	*p=0.046*
Combined (Best-FSB-first)	0.920	*p=0.145*

With more shallow semantic matching techniques in FE-NFES and FE-FES where the overall frame scenarios are ignored, the AP performance decreases quite drastically compared to that of the frame alignment methods. This is because of two issues: i) the FE redundancy among the different frames in FrameNet, and ii) the lack of *semantic normalization*[2] that is achieved by conducting frame matching prior to FE matching. In the absence of such normalization, the FE alignment strategies suffer from the shallow semantic role-based dependencies that miss their meta-semantics in semantic classes. These results suggest that semantic roles (FEs) and semantic classes (frames) are more useful when used in concert as in CFFE, FSFE-NFS, and FSFE-FS where frame matching is followed by a FE matching procedure.

In the 2006 runs, however, the results are very consistent as there is not much change in the MRRs with the different FSB techniques. This is mostly due to the sparse frame and FE assignment in the absence of any manual annotation over the automated annotation by SHALMANESER. In a sparse annotation grid, there is not much evidence for comparing the results achieved via the different FSB methods.

Generally, in our experiments, a frame matching process prior to the FE alignment task is shown to be effective and can significantly affect AP performance. However, in the presence of different challenges which interfere with a high performance complete FE alignment procedure as in CFFE (such as differently assigned FEs and erroneous string matching), a relaxed procedure of FE alignment at this stage is preferred. In addition, in an exhaustively annotated corpus with many frames evoked, a frame scoring strategy for pinpointing the answer spans and ranking the answer candidates is required. Therefore, our FSFE-FS method is selected as the best performing FrameNet-based factoid AP method which outperforms FE-oriented methods and improves previously existing frame and FE-oriented approaches. The FSFE-FS method has shown even higher performance than other FSB approaches (not pure frame alignment techniques) such as those in [11], [7], [8].

To see how this method could impact the best-performing TREC 2004 system, an *artificial* combined processing task is considered. For the time being, there is no automated way of deciding when to use FSB in conjunction with other AP methods (no robust characterization of questions exists); therefore, we combine the results of the best FSB method with those of the best-performing TREC system manually. Results of the two possible combined settings as well as the best-performing TREC

[2] Grouping lexical items which share the same semantic features such as semantic roles.

system in the TREC 2004 track are shown in Table 3. These results show that the combined Best-TREC-first AP method produces significant improvement (with p=0.046) over the AP MRR of the best-performing TREC system.

4 Lexical Coverage

The performance of a QA system which uses the FSB model of AP is dependent on the lexical coverage of FrameNet. In this section, the effect of lexical coverage on AP performance is studied to understand sensitivity of QA systems to different POS predicates.

The FrameNet project is being developed on a frame basis instead of a predicate basis which makes the task of covering English predicates slow. The current standing of FrameNet in this regard can be characterized by:

1- Predicate coverage: which indicates if a lexical entry is covered by any of FrameNet semantic frames, and
2- Sense coverage: indicating if a predicate with a known semantic sense[3] is covered in a FrameNet frame.

There is not much formal information about FrameNet coverage available; however, according to [16], the FrameNet 1.2 dataset covers only 64% of the tokens in the Penn Treebank and 26% of the token types. We have conducted a naïve coverage analysis on parts of the text in the AQUAINT collection from which the answers for the TREC 2004 factoid questions are to be extracted. We explore a *random* list of top 10 passages retrieved for 10 factoid questions in the TREC 2004 track (100 passages in total). This analysis sheds some light on the proportions of coverage of different POS predicates in the FrameNet 1.3 dataset. Table 4 summarizes the statistical information of this sub-collection.

Table 4. A subset of the AQUAINT collection for analysing the FrameNet lexical coverage

Element	Number
Passages from AQUAINT	100
Sentences	233
Single words (all)	6006
Single words (unique)	1611
Predicates (all)	3567
Prediates (unique)	1404

The coverage analysis on this sub-collection measures the number of target predicates which could have been covered as Frame Evoking Elements (FEEs) that could evoke FrameNet frames. From a statistical viewpoint, the minimum number of samples (predicates) required for analyzing the proportions at the confidence level 95% and the margin of error 0.03 (desired precision $\pm$ 0.03) is 1068. Therefore, even the total number of unique occurrences of predicates (1404) suffices for the analysis of population proportions. Table 5 depicts the number of the predicates which are not covered after the task of manual annotation using the FrameNet 1.3 dataset.

[3] For instance, the predicate "make" has different senses such as cooking, and creation.

The first column titled "Overall" in Table 5 shows the values acquired when taking into account all the sentences at once as a unique set. The "Avg." column, however, includes the values obtained as average values over 10 sets of sentences. These results are all showing both the predicate and sense coverage together. A POS-based analysis of not-covered predicates has also been conducted to observe more detailed rates of lack of coverage over different POS predicates. The result is shown in Figure 6. It is observed that the majority of the predicates in our analysis sub-collection which have not been covered in FrameNet are nouns.

Table 5. Target predicates not-covered after manual correction with FrameNet 1.3

Element	Overall		Avg.	
	all	unique	all	unique
Not-covered predicates	1014	528	101.4	61.7
Normalized by sentences	4.351	2.266	4.348	2.711
Normalized by words	0.168	0.286	0.162	0.234
Normalized by predicates	0.284	0.376	0.274	0.325

Before conducting the AP experiments, we have extracted some FrameNet statistics from the two FrameNet 1.2 and 1.3 datasets summarized in Table 6. In general, the progress in covering prepositions and verbs is better than the other POS predicates. Noun predicates still need some more effort whilst the coverage of adverbs and adjectives seems in a weaker situation that requires more work.

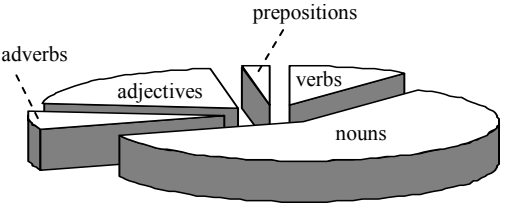


Fig. 6. POS-based analysis of *all* not-covered predicates in FrameNet 1.3

There are two issues that can affect the results of AP in the sense of FrameNet coverage in each dataset: i) the SHALMANESER classifiers which can be trained with the two FrameNet 1.2 and 1.3 datasets, and ii) the manual annotation correction process which can be considered with any of these datasets.

For the TREC 2004 test set, the effect of coverage analysis is performed on the basis of SHALMANESER trained with the FrameNet 1.2 dataset and corrected with the two FrameNet datasets. For the TREC 2006 test set there is no corrected annotation data available. Therefore, the effect is measured with the different SHALMANESER instances trained with different FrameNet datasets. Table 7 summarizes the results using the FSFE-FS method of AP.

The first observation from the experimental results in Table 7 is that in an effectively annotated environment, there is a higher chance of retrieving more correct factoid answers for the AP module as the coverage ratio of predicates in FrameNet grow.

Table 6. Statistical information of the two FrameNet 1.2 and 1.3 datasets

Figure	Predicates	Verbs	Nouns	Adj.	Adv.	Prep.
FN 1.2 dataset	8755	3424	3673	1536	39	72
FN 1.3 dataset	9454	3891	3730	1680	49	91
Growth ratio	7.984%	13.639%	1.551%	9.375%	25.641%	26.388%

Table 7. QA runs with different FrameNet datasets for: annotation correction (2004) and SHALMANESER training (2006)

FrameNet dataset	MRR	
	TREC 2004	TREC 2006
FN 1.2	0.600	0.011
FN 1.3	0.627 *p=0.369*	0.006 *p=0.281*

The improvement in the QA performances with different lexical coverage rates in FrameNet, however, is not statistically significant at this time which is due to the low ratios of progress in coverage in FrameNet 1.3 compared to FrameNet 1.2.

By focusing on the results on the TREC 2006 dataset, it is inferable that in a sparsely annotated text collection, a higher predicate coverage may even damage the QA performance. This is because in a sparse and inaccurate annotation environment, resulted by an inaccurate automated parser, there is more possibility for extracting wrong answers by a larger number of wrongly assigned frames and FEs.

After our analysis on the coverage of different POS predicates, it is shown that noun predicates are covered less than all other POS predicates (*R1*). Intuitively, it is accepted that in finding answers to factoid questions, verb and noun predicates play more important roles (*A1*). Their significance is due to the fact that the main actions of question events are more associated to the verbs and nouns in questions. The induced growth ratio in terms of verbs (13.639%) in FrameNet is more promising than that of nouns (1.551%) (*A2*). As a result of ((*R1*) AND (*A1*) AND (*A2*)), the work on covering more noun predicates in FrameNet is concluded to be more important at this stage to make a good balance in coverage rates in the next releases of FrameNet. This signals greater potential for factoid QA systems to extract more correct answers.

5 Concluding Remarks

We have studied different methods of Answer Processing (AP) using FrameNet frames and Frame Elements (FEs) alignment. After implementing five different methods of frame semantic alignment for AP, it has been shown that our FSFE-FS method based on the relaxed matching of frames and FEs (no necessity for aligning all FEs) with the query contexts in answer-bearing passages results in the highest AP performance and improves existing Frame Semantic-Based (FSB) methods of AP.

We have performed some analyses on the lexical coverage of FrameNet over English predicates and conducted some experiments which show how the lexical coverage issue may affect the overall performance of AP. A low coverage can reduce the number of questions that can be answered in two ways; i) the main predicate of the question is not covered and there will be no way of getting the answer using frame and FE

alignment, and ii) the answer sentence and/or its predicate is not covered in the passages and again the answer cannot be identified. We have shown that, in order to improve AP performance in FSB models, the work on covering more *noun* predicates in FrameNet is crucial at this stage, while the coverage rate and outlook of the other important part-of-speech predicates – verbs – is more reasonable so far.

References

1. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The Berkeley FrameNet project. In: International Conference on Computational Linguistics (1998)
2. Fillmore, C.J.: Frame semantics and the nature of language. In: Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech, vol. 280, pp. 20–32 (1976)
3. Petruck, M.R.L.: Frame semantics. In: Verschueren, J., et al. (eds.) Handbook of Pragmatics 1996. John Benjamins, Philadelphia (1996)
4. Narayanan, S., Harabagiu, S.: Question answering based on semantic structures. In: International Conference on Computational Linguistics (COLING 2004), Switzerland (2004)
5. Fliedner, G.: Deriving FrameNet representations: towards meaning-oriented question answering. In: Meziane, F., Métais, E. (eds.) NLDB 2004. LNCS, vol. 3136. Springer, Heidelberg (2004)
6. Kaisser, M.: QuALiM at TREC 2005: web-question answering with FrameNet. In: The Fourteenth Text Retrieval Conference, TREC 2005 (2005)
7. Kaisser, M., Scheible, S., Webber, B.: Experiments at the University of Edinburgh for the TREC 2006 QA track. In: The Fifteenth Text Retrieval Conference, TREC 2006 (2006)
8. Kaisser, M., Webber, B.: Question answering based on semantic roles. In: The ACL 2007 Deep Linguistic Processing Workshop (ACL-DLP 2007), Prague, Czech Republic (2007)
9. Hickl, A., et al.: Question answering with LCC's CHAUCER at TREC 2006. In: 2006 Text Retrieval Conference, TREC 2006 (2006)
10. Ofoghi, B., Yearwood, J., Ghosh, R.: A hybrid question answering schema using encapsulated semantics in lexical resources. In: Sattar, A., Kang, B.-h. (eds.) AI 2006. LNCS (LNAI), vol. 4304. Springer, Heidelberg (2006)
11. Shen, D., Lapata, M.: Using semantic roles to improve question answering. In: 45th Annual Meeting of the Association for Computational Linguistics, Prague (2007)
12. Ofoghi, B., Yearwood, J., Ma, L.: The impact of semantic class identification and semantic role labeling on natural language answer extraction. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) ECIR 2008. LNCS, vol. 4956, pp. 430–437. Springer, Heidelberg (2008)
13. Clarke, C., Cormack, G., Tudhope, E.: Relevance ranking for one to three term queries. In: 5th International Conference Recherche d'Information Assistee par Ordinateur, RIAO 1997 (1997)
14. Ofoghi, B., Yearwood, J., Ghosh, R.: A semantic approach to boost passage retrieval effectiveness for question answering. In: 29th Australian Computer Science Conference (2006)
15. Erk, K., Pado, S.: Shalmaneser - A toolchain for shallow semantic parsing. In: LREC 2006 (2006)
16. Honnibal, M., Hawker, T.: Identifying FrameNet frames for verbs from a real-text corpus. In: Australasian Language Technology Workshop 2005, Sydney, Australia (2005)

Learning to Find Relevant Biological Articles without Negative Training Examples

Keith Noto, Milton H. Saier Jr., and Charles Elkan

University of California, La Jolla CA 92093
knoto@cs.ucsd.edu,msaier@biomail.ucsd.edu,elkan@cs.ucsd.edu

Abstract. Classifiers are traditionally learned using sets of positive and negative training examples. However, often a classifier is required, but for training only an incomplete set of positive examples and a set of unlabeled examples are available. This is the situation, for example, with the Transport Classification Database (TCDB, www.tcdb.org), a repository of information about proteins involved in transmembrane transport. This paper presents and evaluates a method for learning to rank the likely relevance to TCDB of newly published scientific articles, using the articles currently referenced in TCDB as positive training examples. The new method has succeeded in identifying 964 new articles relevant to TCDB in fewer than six months, which is a major practical success. From a general data mining perspective, the contributions of this paper are (i) evaluating two novel approaches that solve the positive-only problem effectively, (ii) applying support vector machines in a state-of-the-art way for recognizing and ranking relevance, and (iii) deploying a system to update a widely-used, real-world biomedical database. Supplementary information including all data sets are publicly available at www.cs.ucsd.edu/users/knoto/pub/ajcai08.

1 Introduction

The transport classification database, or TCDB (www.tcdb.org), is an online database which contains sequence, structural, and functional information about proteins that relate to transport across cell membranes in a variety of organisms, categorized into over 550 families of proteins [11]. TCDB is widely used, averaging over 50 different users per day from research institutions all over the world. TCDB defines and implements the transport classification system [10] for categorizing transport proteins, which was adopted by the International Union of Biochemistry and Molecular Biology as the international standard in 2002.

As of October 15, 2007, the start of the project described in this paper, the data contained in TCDB were compiled from 3,403 publications in over 200 different journals. Our goal is to help keep TCDB updated with all relevant new information related to transport proteins. The sources of information that we consider in this paper are recent articles published in biological journals. Information that goes into TCDB is verified by a human expert before inclusion. Accordingly, our goal is not full automation. However, even if we restrict the set

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 202–213, 2008.

of journals to those already referenced in TCDB, there are several thousand articles published each month—too many for a human expert to examine. Of these, only a small percentage are relevant to TCDB. Our goal is to identify these relevant articles automatically, as soon as possible after their abstracts are published in Medline.

The procedure for providing the expert with candidate articles is to (i) learn a model for ranking future articles, (ii) use the model to rank articles from a candidate set, and (iii) provide the expert with as many articles as the expert is able to screen (we call this number k—typically a small percentage of the candidate articles). Therefore, the technical task we consider in this paper is to learn a classifier for Medline abstracts that maximizes the number of relevant articles in the set of size k shown to the human expert.

Of course, this task is not unique to TCDB. The *Nucleic Acids Research* journal lists over 1,000 specialized databases as of the January, 2008 update [6]. With over 50,000 citations currently added to Medline each month, the need for methods to screen and select documents automatically is increasing. Similar tasks have been investigated before [4, 17, 19], but previous approaches assume the availability of labeled negative examples as well as of labeled positive examples. Below, we investigate the implications of only having positive and unlabeled training examples available.

We use a support vector machine (SVM) as our classifier. The available data for training consist of (i) articles already referenced in TCDB and (ii) recent articles published in the same journals. In principle, all 18 million Medline documents could be unlabeled training examples, but in practice we use a sample of these that is limited to recent articles from journals known to contain relevant articles. Articles in the first training set are labeled positive, but articles in the second set are unlabeled. Thus, we face the "positive-only" issue: We wish to learn a classifier that discriminates positives from negatives, but we have no specific negative training examples. We do however know that most of the unlabeled examples are negative.

Our human expert is able to screen only a limited number of documents. Given a set of unclassified abstracts (*i.e.* a test set of future articles), our classifier scores them according to their likelihood of being relevant to the TCDB database. We then deliver the highest-scoring ones to the human expert for screening. We wish to minimize the the number of delivered articles that turn out to be irrelevant. Thus the objective function to maximize is not classification accuracy on the whole test set, but rather precision on a small high-scoring subset of the test set.

Formally, let x represent the features of an article, and let $y = 1$ represent the fact that the article is relevant (in this case, to TCDB). We wish to learn a scoring function $f(x)$ that, given a set of examples $T = \{x_1, x_2, ..., x_n\}$, maximizes

$$precision(T, f, k) = \frac{1}{k} \sum_{i=1}^{n} I(rank(x_i, f) \leq k \wedge y_i = 1) \tag{1}$$

where the indicator function $I(\cdot)$ returns 1 if its argument is true, 0 otherwise, and $rank$ gives the position of x_i if T is sorted by descending $f(x)$; that is,

$rank(\arg\max_x f(x), f) = 1$. In words, Equation (1) gives the proportion of examples in the top k according to f that are positive, which is called the precision at k. This is our objective function and is therefore used throughout this paper to evaluate our models.

The fact that we are given only positive and unlabeled examples at the outset from which to learn f is the main difference between ours and previous approaches which learn from positive and negative training examples [19,4,17]. Our approach and previous approaches to similar tasks, which learn naïve Bayes [19], and SVM models [4,17], as well as nonstandard classifiers [3,4] from positive and negative training examples.

In Section 2, we describe how to train an SVM to achieve good classification accuracy. In Section 3, we explain two solutions to the positive-only issue. The first solution iteratively retrains a classifier, while the second solution relies on a new mathematical result. In Section 4, we describe how we estimate the recall of our models without negative examples. In Section 5, we describe how we the final trained classifier has been deployed successfully to find hundreds of new relevant articles for TCDB.

2 Data and Training

The data set that we use to develop and evaluate our methods consists of:

- 3,403 articles that appear in Medline and were referenced in TCDB as of October 15, 2007–these are our positive examples, and
- 16,341 unlabeled articles published recently in those journals referenced in TCDB.

The abstracts of these articles are obtained using the following procedure: First, we download the article data using NCBI's entrez programming utilities.[1] The positive examples are downloaded individually using their PubMed ID number.[2] The unlabeled examples are downloaded using the query term "JOURNAL [TA] & MINDATE= ... & MAXDATE= ..." for each journal in TCDB, for the date ranges of October 1 to 31, 2007 (retrieved on December 12, 2007) and November 1 to December 20, 2007 (retrieved on December 20, 2007). This results in a subset of the articles in the PubMed ID range 17902656 to 18092361. The subset contains 16,341 articles because it is restricted to the articles that appear in certain journals and have PubMed dates in the given range associated with them. We represent each article as a vector of features which are to the log-scaled counts of word stems that appear in the article's abstract. We represent each article as a vector of features corresponding to the words in the article's abstract. We apply the Porter stemming algorithm [14] to the abstract text, count the use of each word in each abstract using MALLET [13], and limit the vectors to words

[1] See `eutils.ncbi.nlm.nih.gov/entrez/query/static/esearch_help.html`.

[2] All Medline documents have a unique ID number in PubMed, which is NCBI's public interface to Medline.

that appear at least three times in the corpus. This reduces the vocabulary size in our data set by 47% (17,060 words appear once, 6,534 words appear twice, and we retain a vocabulary of 26,364 words that appear at least three times). Finally, we represent each word by the value $\log(1+n_i)$, where n_i is the number of times word i appears in an abstract, and we normalize the vector for each article to have unit Euclidean length, *i.e.* to lie on the unit sphere. We do not use information associated with the articles (journal names, author names, *etc.*) other than the abstract text. We verify that the vocabulary does not contain any obvious "leaker" words, *i.e.* terms that discriminate between the training subsets but are not valid predictors for future articles. In particular, the trained models do not use article dates, directly or indirectly, to make predictions. Recall the evaluation metric in Equation (1). For this data set, we choose $k = 3000$, this being about the number of articles the human expert is able to screen in the time available, considering that the positive set is already labeled.

We label training examples that are obtained from TCDB as positive, and we tentatively label the other examples as negative, even though some of them must actually be positive. Based on an informal examination of the articles during the course of this research, we estimate this fraction at about 2.3%.

We use a support vector machine (SVM) as our classification method. Specifically, we use the *svm-light* implementation of soft-margin SVMs [8].Alternatives to SVMs include maximum entropy and naïve Bayes models. One reason that we choose to use SVMs is because they are a discriminative (as opposed to generative) method. Maximum entropy is also discriminative, but SVMs have two additional advantages. First, their training objective is to maximize the margin in feature space between positive and negative examples, which gives them a small but useful boost in accuracy compared to methods like maximum entropy that maximize variants of log likelihood instead. Second, SVMs facilitate the use of nonlinear kernels. There are methods for learning SVMs that directly optimize nonlinear performance measures like Equation (1) [9], but for simplicity we use standard SVMs. We do not, however, use the SVM's natural threshold directly. We use the SVM to generate a ranking of examples, and we deliver the top-ranked k examples to the human expert. To evaluate trained classifiers fairly, we use ten-fold cross-validation. For each of the ten test folds, we deliver the highest-scoring 300 articles (for a total of $k = 3000$) to the human expert for inspection.

We train all SVMs using a quadratic kernel. We choose this kernel because it allows interactions between words to influence predictions, in addition to individual words. In preliminary informal experiments, it performs as well as or better than other standard kernels on our task. There is no known principled way to select an appropriate value for the SVM soft-margin penalty parameter C (the tradeoff between margin width and training set accuracy). The optimal value of C for our task may be different from its optimal value in other tasks, because some of the unlabeled examples are relevant to TCDB and will be misclassified when tentatively labeled as negative. Therefore, to select C we use nested cross-validation (*i.e.* tuning set cross-validation within testing set cross-validation). We search for the best value of C by starting at $C = 1$, and trying both higher

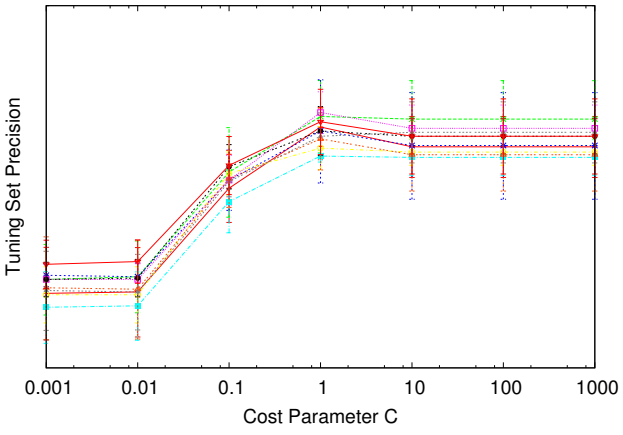


Fig. 1. The tuning set precision associated with the value of the cost parameter in our SVM models. Each line corresponds to one of ten folds.

and lower values until the tuning set precision begins to decrease. To evaluate each value of C, we use four-fold cross-validation and consider powers of 10. Figure 1 shows the tuning set precision for each of ten testing folds. Our approach chooses $C = 1$ in nine of ten folds, and $C = 10$ once.

3 Learning without Negative Examples

As explained above, a major difficulty of our task is that we do not begin with any set of examples known to be genuinely negative. This difficulty is not unique to our task: In many application domains only positive examples are available naturally for training. Often, negative examples in these domains are ubiquitous, but it is too time-consuming or costly to label and organize them explicitly. The issue of learning a classifier from unlabeled and positive training examples, instead of from negative and positive examples, has been investigated before, *e.g.* [15, 12, 2, 16, 18]. However, the issue has not yet been investigated in the context of the task of identifying articles that are relevant to a biomedical database. We provide a comparison with the aforementioned methods in [5], and we believe that our recent formalization (explained in detail Section 3.2) demonstrates that our approaches are well-suited to this task.

3.1 Iterative Relabeling

Our first approach to this issue is to find genuine labels for unlabeled examples predicted to be positive, following an iterative technique introduced by Das *et al.* [1] for the application of identifying relevant database records. That is, we show the unlabeled portion of the top k predictions to a human expert, who assigns each article to one of the following categories:

– Accepted (and added to TCDB)
– Not relevant (the article is not about transport proteins)
– Relevant, but not accepted

The "relevant, but not accepted" category contains articles that are about transport proteins, but that contain information already in TCDB from a different article, or information of low importance and therefore excluded at the discretion of the human expert. Examples in these three categories are labeled positive, labeled negative, and discarded, respectively. After relabeling, we learn a new SVM model and repeat the process.

We perform this procedure until there are no unlabeled predictions in the top k, or until a fixed number of iterations has been reached. The human expert has fewer articles to screen each time, because most of the top k are the same articles each time. Table 1 shows how many examples fall in each category during the relabeling process. Note that fewer and fewer articles that appear in the top k are marked as "not relevant" to TCDB. Very few of the remaining unlabeled articles are likely to be relevant to TCDB. For this reason, we did not ask the human expert to spend the time to do more than two iterations.

Table 1. The number of unlabeled examples that fall into the top k predictions and the assignments given to them by the human expert during each iteration of relabeling

Iteration	Unlabeled Examples in Top k	Accepted into TCDB	Not Relevant	Relevant, but not Accepted
1	386	182	126	78
2	49	18	16	15

Although the human expert makes a distinction between accepting an article to be added into TCDB and classifying it as relevant, but not accepted, we do not attempt to distinguish between these two article types automatically. The expert makes this distinction because the task is to find new and important information for TCDB. However, articles of both types are indeed about transport proteins. Many relevant but rejected articles are important, and are not accepted only because they contain information that is already present in TCDB.

3.2 Transforming Predicted Relevance Scores

The previous section explains an iterative process that uses a human expert to provide the true labels for unlabeled training examples that are tentatively predicted to be relevant. This process does improve the accuracy of the final classifier obtained, but only slightly. In this section, we employ a recent mathematical result to explain why the improvement is slight [5].

Let x represent an article in the entire Medline universe, and let s be a random variable such that $s = 1$ means that the article x is selected as a positive training example, *i.e.* x is one of the articles cited currently in TCDB. Let $y = 1$ mean

that the article x is truly relevant to TCDB, whether or not x is currently cited in TCDB. Our goal is to find precisely those x for which $y = 1$ but $s \neq 1$.

Assume that every relevant article in the universe has an equal chance of being included in TCDB already. Mathematically, this assumption is that $P(s = 1|y = 1, x)$ is a constant. Of course, newly published articles in the universe cannot have an equal chance of being already in TCDB. However, since our methods do not use dates in any way, this assumption is reasonable when x refers to only the utilized characteristics of an article.

The goal is to learn a function $f(x)$ such that $f(x) = P(y = 1|x)$ as closely as possible. Suppose we provide the labeled data ($s = 1$) and unlabeled data ($s = 0$) as inputs to a standard training algorithm. This algorithm will yield a function $g(x)$ such that $g(x) = P(s = 1|x)$ approximately. The following lemma shows that $f(x)$ is a transformed version of $g(x)$.

Lemma 1. Suppose $P(s = 1|y = 1, x)$ is a constant. Then $P(y = 1|x) = P(s = 1|x)/P(s = 1|y = 1)$.

Proof. The assumption that $P(s = 1|y = 1, x)$ is a constant implies that $P(s = 1|y = 1, x) = P(s = 1|y = 1)$ for all articles x. So

$$\begin{aligned} P(s = 1|x) &= P(y = 1 \wedge s = 1|x) && \textit{(because } s = 1 \textit{ implies } y = 1\textit{)} \\ &= P(y = 1|x)P(s = 1|y = 1, x) && \textit{(rewritten)} \\ &= P(y = 1|x)P(s = 1|y = 1) && \textit{(assumption)} \end{aligned}$$

The result follows by dividing both sides by $P(s = 1|y = 1)$. ∎

Although the proof above is simple, this result was published only recently [5]. The reason that the result is novel is perhaps that although the scenario of learning from unlabeled and positive examples has been discussed in many previous papers, it has not previously been formalized using the s random variable.

The lemma says that $P(y = 1|x)$ is a constant times $P(s = 1|x)$. Hence, sorting examples x by their predicted value $P(s = 1|x)$ gives the *same ranking* as sorting by $P(y = 1|x)$. Suppose we use an SVM to sort examples. Although the scores given by such a classifier are not correct probabilities $P(s = 1|x)$, the best estimators of these probabilities are monotonically increasing functions of the SVM scores. Hence, sorting examples by their SVM scores also in principle gives the same ranking as sorting examples by $P(y = 1|x)$.

Given the argument above, the top k abstracts identified by an SVM trained on the unlabeled and positive training examples should be the same as the top k identified by an SVM trained on negative and positive examples. In other words, relabeling should have no effect on the outcome of the model.

Table 2 shows how our classifier improves during the relabeling process. The numbers in this table are computed using knowledge of positive and negative labels acquired via the relabeling process. Comparing lines in the table shows that the relabeling process only leads to a small increase in success (indeed, the results of the second iteration are slightly worse). The lemma above can be viewed as an explanation of why this increase is small. It implies that if they have limited resources, future researchers faced with a database curation task similar

Table 2. Distribution of the top-k predictions for multiple iterations of relabeling

Iteration	Relevant	Not Relevant
Before Relabeling	2873	127
After 1 Iteration	2898	102
After 2 Iterations	2896	104

to the addressed in this paper will not lose much by not identifying genuine negative training examples manually.

4 Estimating Recall

Although we can estimate the precision of our approach by examining the labels that the human expert assigns to the high-scoring articles predicted by our model, we cannot estimate the *recall* of our models without knowing which of the unlabeled test set examples are false negatives.

To estimate the recall of our models, we take advantage of a separate set of relevant articles that are chosen by the human expert directly from the literature. We then assume that all of the articles from those same journal issues which are *not* selected by the human expert are genuine negative examples. The set of positive examples consists of 37 articles from a total of 44 issues of three particular journals, *the Journal of Bacteriology*, *the Journal of Biological Chemistry*, and *Nature*, between the dates of September 7 and December 14, 2007. The set of negative examples consists of 370 articles from the same journal issues. We train our model on a set of positive articles that are referenced by TCDB, and a set of unlabeled articles from the same journals referenced by TCDB, all of which have a PubMed date strictly prior to September 7, 2007.

Results are shown in Figure 2. It is difficult to decide an appropriate value for the threshold k to use in this figure. We choose $k = 39$ based on the following reasoning. The 418 articles from three journals span approximately a period of time when we delivered 600 articles to our human expert from a total of 102 journals (the three mentioned above, and 99 more). So, we use our learned model to rank all 15,125 articles from these 102 journals in this time period. We find that of the top 600 predictions across all journals, $k = 39$ concern papers in the three journals of interest. In detail, the 600 predictions break down as follows:

	Positive examples from three journals	Negative examples from three journals	Unlabeled examples from 99 journals
$rank \leq k$	21	18	561
$rank > k$	16	363	14,146

Based on the threshold $k = 39$, our model recovers about 57% of the articles found in the literature. However, a much higher recall, up to 90%, is achievable with reasonable increase in human effort via a larger k. Moreover, 57% recall is acceptable because much relevant information is published more than once,

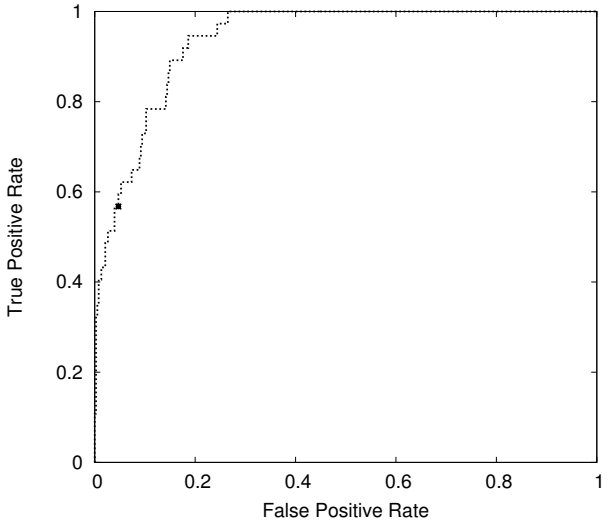


Fig. 2. An ROC curve showing the performance of our classifier on 418 articles from three journals, 37 of which were manually identified as relevant to TCDB. The remaining 381 articles are from the same journal issues and are considered negative examples. The point highlighted on the curve corresponds to a reasonable number of articles to show to the human expert.

and/or can be found by following citation paths starting at papers that are detected.

In the course of further research the human expert re-evaluated 25 of the articles from the 44 issues of the three journals that he dismissed previously. He then decided that 11 of these *are* relevant. This inconsistency indicates that the precision and recall of human procedures are far from perfect. Overall, the automated procedure has recall comparable to that of a human, and is definitely helpful in identifying articles that would otherwise be missed simply by browsing the literature. Indeed, at $k = 39$ our procedure recovers 8 of the additional 11 articles that the human found upon re-evaluation.

5 Deployment

We measure how effective our approach is in deployment by applying the classifier trained on the relabeled data set to a new data set. To do this, we download a new set of articles, again restricted to the same journals already referenced in TCDB, but now appearing in PubMed after the articles in the training set.

During these experiments, we generated five new data sets. These are produced when the human expert requests them, so they vary in size. For each set, we show the $k = 300$ highest-scoring articles to the human expert. After each set is deployed, the human expert screens and labels the top k. This means that we

Table 3. The labels given by the human expert to the top 300 articles identified by our trained classifier on five datasets spanning a period of approximately October 1, 2007 to March 13, 2007. The rightmost column indicates the proportion of the deployment sets which are true and false positives.

Approximately 4 weeks, October 1 - October 31 2007, 6108 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	62	30	17	109 (1.78%)
Labeled relevant, but not accepted	12	12	12	36 (0.59%)
Not relevant	26	58	71	155 (2.53%)

Approximately 7 weeks, November 1 - December 20 2007, 10233 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	54	33	9	96 (0.94%)
Labeled relevant, but not accepted	13	3	9	25 (0.24%)
Not relevant	33	64	82	179 (1.75%)

Approximately 3 weeks, December 21 - January 15 2008, 3885 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	50	20	9	79 (2.03%)
Labeled relevant, but not accepted	28	21	10	57 (1.47%)
Not relevant	22	59	81	164 (4.22%)

Approximately 5 weeks, January 15 - February 20 2008, 6975 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	51	27	8	86 (1.23%)
Labeled relevant, but not accepted	28	39	20	87 (1.25%)
Not relevant	21	34	72	127 (1.82%)

Approximately 3 weeks, February 21 - March 13 2008, 2544 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	24	13	6	43 (1.69%)
Labeled relevant, but not accepted	38	19	8	65 (2.56%)
Not relevant	38	68	86	192 (7.55%)

have access to an increasing number of training labels. After each deployment set is labeled, the classifier is retrained using the new labels.

Table 3 shows the disposition of these top 300 articles, as assigned by the expert, in each of the five deployment sets. As expected, the precision of our models decreases with the scoring rank of the article: the top 100 articles are more likely to be genuinely relevant than the articles ranked 100-200 or 200-300. The proportion of relevant documents varies from 1.18% (deployment set 2, November and December 2007) to 4.25% (deployment set 5). It is not obvious what the source of this variability is. The source may be underlying variability in the content of the journals, and it may be variation in the standards of the human expert, who is influenced by the overall quality and quantity of the deployment sets. Note that the proportion of articles accepted into TCDB varies less, ranging

between 0.94% (deployment set 2) and 2.03% (deployment set 3), indicating that our approach is able to provide a consistent stream of important documents.

6 Conclusion

We have shown four results that we believe are interesting and of general significance. First, SVMs can be used effectively as a method of ranking text documents according to likely relevance to a specialized biomedical database, and not just as a method of yes/no classification. Second, our method for iteratively relabeling examples is an efficient and effective approach to the positive-only issue. Third, this issue is in fact less important than it appears, because of the equivalence established in Lemma 1. Fourth, our methods can successfully provide an up-to-date stream of highly relevant documents to the human maintainers of TCDB, a widely-used protein database. In total, these methods have so far discovered 964 relevant articles, 626 of which have been added to TCDB. This represents a 16% increase in the number of references in TCDB over the last six months, which is much faster than the previous rate of growth of TCDB.[3]

Now that we have established that our approach is useful for updating real databases like TCDB, we plan to apply our learned classifiers to papers published outside the set of journals already known to contain some relevant articles, in order to find relevant articles that would otherwise certainly be missed. We also plan to extend our feature set. In particular, Wang *et al.* [17] show that author names, medical subject headings (MeSH) and standardization of biological strings (*e.g.* converting "9-mer" to "x-mer") can improve performance. Han *et al.* [7] hypothesize that a characteristic of biomedical text is that it contains informative suffixes and show that using substrings as features can outperform general-purpose stemming algorithms, such as the Porter stemmer, when classifying biomedical text. In addition, we plan to incorporate journal identity, title words, paper length, paper category (*e.g.* based on keywords or as curated by Medline), and paper type (*e.g.* review) as document features.

Recall that *Nucleic Acids Research* lists over 1,000 specialized databases. Although our experiments to date focus exclusively on only one of them, our methods are directly applicable to many more of these databases.

Acknowledgments. This research is funded by NIH grant GM077402.

References

1. Das, S., Saier Jr., M.H., Elkan, C.: Finding transport proteins in a general protein database. In: Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 54–66 (2007)
2. Denis, F., Gilleron, R., Letouzey, F.: Learning from positive and unlabeled examples. Theoretical Computer Science 348(1), 70–83 (2005)

[3] Most of these articles were discovered during the five deployment sets described in Section 5, but some came from the iterative relabeling experiments described in Section 3 and some came from various experimental trials not described above.

3. Dobrokhotov, P.B., Goutte, C., Veuthey, A.L., Gaussier, E.: Combining NLP and probabilistic categorisation for document and term selection for Swiss-Prot. In: Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology, pp. 91–94 (2003)
4. Dobrokhotov, P.B., Goutte, C., Veuthey, A.L., Gaussier, E.: Assisting medical annotation in Swiss-Prot using statistical classifiers. International Journal of Medical Informatics 74(2-4), 317–324 (2005)
5. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), pp. 213–220 (2008)
6. Galperin, M.Y.: The Molecular Biology Database Collection: 2008 update. Nucleic Acids Research, 36(Database issue):D2 (2008)
7. Han, B., Obradovic, Z., Hu, Z., Wu, C.H., Vucetic, S.: Substring selection for biomedical document classification. Bioinformatics 22(17), 2136–2142 (2006)
8. Joachims, T.: Making large-scale support vector machine learning practical. In: Smola, A., Schölkopf, B., Burges, C. (eds.) Advances in Kernel Methods: Support Vector Machines. MIT Press, Cambridge (1998)
9. Joachims, T.: A support vector method for multivariate performance measures. In: ACM International Conference Proceeding Series, vol. 119, pp. 377–384 (2005)
10. Saier Jr., M.H.: A functional-phylogenetic classification system for transmembrane solute transporters. Microbiology and Molecular Biology Reviews 64(2), 354–411 (2000)
11. Saier Jr., M.H., Tran, C.V., Barabote, R.D.: TCDB: The transporter classification database for membrane transport protein analyses and information. Nucleic Acids Research 34, D181–D186 (2006)
12. Liu, B., Dai, Y., Li, X., Lee, W.S., Yu, P.S.: Building text classifiers using positive and unlabeled examples. In: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), pp. 179–188 (2003)
13. McCallum, A.K.: MALLET: A machine learning for language toolkit (2002), `http://mallet.cs.umass.edu`
14. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
15. Steinberg, D., Cardell, N.S.: Estimating logistic regression models when the dependent variable has no variance. Communications in Statistics - Theory and Methods 21(2), 423–450 (1992)
16. Wang, C., Ding, C., Meraz, R.F., Holbrook, S.R.: PSoL: A positive sample only learning algorithm for finding non-coding RNA genes. Bioinformatics 22(21), 2590–2596 (2006)
17. Wang, P., Morgan, A.A., Zhang, Q., Sette, A., Peters, B.: Automating document classification for the immune epitope database. BMC Bioinformatics 8(269) (2007)
18. Ward, G., Hastie, T., Barry, S., Elith, J., Leathwick, J.R.: Presence-only data and the em algorithm. Biometrics (2008)
19. Wilbur, W.J.: Boosting naïve Bayesian learning on a large subset of MEDLINE. In: Proc. AMIA Symp. (2000)

Humor Prevails!
- Implementing a Joke Generator into a Conversational System

Pawel Dybala, Michal Ptaszynski, Shinsuke Higuchi, Rafal Rzepka, and Kenji Araki

Graduate School of Information Science and Technology, Hokkaido University
Kita 14 Nishi 9, Kita-ku, 060-0814 Sapporo, Japan
{paweldybala,ptaszynski,shin_h,kabura,araki}@media.eng.hokudai.ac.jp

Abstract. This paper contains the results of evaluation experiments conducted to investigate if implementation of a pun generator into a non-task oriented talking system improves the latter's performance. We constructed a simple joking conversational system and conducted one user evaluation experiment and two third person evaluation experiments. The results showed that humor does have a positive influence on the dialogue between humans and computers. The implications of this fact and problems that occurred during the research are discussed. We also propose how they can be solved in the future.

Keywords: natural language processing, humor, jokes, Japanese puns, pun generator, non-task oriented conversational system.

1 Introduction

1.1 Computing Humor

Even in modern AI, humor processing is still a heavily neglected domain. Also in NLP (natural language processing), research projects on joke generation and recognition are often said to be not really relevant to the main tendencies of the field – which, in fact, does not seem to be proper, as some experiments proved that humor may have a very positive influence on language using machines. For example, experiments conducted by John Morkes et al. [1] proved, that humor enhances task-oriented dialogues, in CMC (computer-mediated communication) as well as in HCI (human-computer interaction). Also results described in this paper (see below) show, that we do need talking engines that would be able to generate humor.

Providing that jokes are "products" of humor (a commonsense definition), it can be stated, that, considering the NLP methodology, the most "computable" genre of jokes is "puns". They can be found in most existing languages – in some of them, however, puns are easier to create and thus their amount is much bigger than in others. One such language is Japanese, in which puns (called *dajare*) are one of main humor genres. This makes Japanese a perfect surrounding for pun processing research. However,

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 214–225, 2008.

although some attempts of constructing pun generating engine have been made, also in Japanese, creating a funny joking conversational system still remains an unfulfilled challenge for NLP researchers.

1.2 Talking for Fun

Nowadays, our interaction partners often are made of silicon and wires – in fact, the relations between us and them often become the subject of scientists' interest. One of the theories (backed up with very robust research) says, that people respond to computers using the same social attitudes and behaviors they apply to other people (SRCT – Social Responses to Communication Technologies theory) [2]. Assuming this to be generally true, if we desire our partners to be able to talk in a free way, also computers should be capable of performing such behaviors. Therefore, construction of systems talking in such non-task oriented manner is supposed to be one of the major goals of today's AI. However, this field is still heavily neglected and requires more research.

Using humor to improve chatbot's performance has already been suggested by Binsted [3], and an attempt of combining joking system JAPE [4] with a talking system Elmo (which, in fact, is not completely non-task oriented system – see reference for details) was made by Loehr [5]. The results were relatively poor, for there was barely any relevance between user's input and system's humorous output. Thus, construction of a funny, humor-equipped talking system still requires much effort and there is very much to be done in this field.

2 Our Two Systems

This research is aimed to investigate if humor really does have a positive influence on non-task oriented conversational system. To do that, we combined our joke generator PUNDA Simple - a simplified version of PUNDA Japanese pun generating system [6] - with a conversational system Modalin created by Higuchi [7]. The combined version of these two systems was named "Pundalin" and used as a main system in our experiment.

The reason for simplifying the pun generating engine was that we wanted to check if humor can improve the human-computer non-task oriented dialogue, in a possibly simple way. If the experiment proved that humor has bad influence on the dialogue or is irrelevant, construction of a more complex system would be pointless. We also wanted to avoid time losses in system's response generation, for in more sophisticated pun generation algorithms, the computation process is very time consuming. Although we agree that this problem has to be solved, in this experiment we assumed that waiting for the system's answer would make user bored, which in turn would probably affect his evaluation of the system.

2.1 Modalin

The baseline system we used in our research was Modalin non-task oriented keyword-based conversational system, developed by Higuchi et al. [7]. The system automatically extracts sets of words related to a conversation topic set freely by a user. The

extraction is based on keywords spotted in user utterances. Than, Goo [8] (in previous version – Google) snippets are used to extract word associations in real time without using earlier prepared resources, such as off-line databases. After the extraction process, the system generates an utterance by adding modality to the retrieved proposition, and verifies the semantic reliability of the proposed sentence. Evaluation experiment showed that over 80% of the extracted word associations were correct. For the details, see reference [7]. Modalin was also used as a "base" for creating a joking system Pundalin – see section 2.3 for details.

2.2 PUNDA Simple

We developed this simple system as a part of PUNDA research project [6], aimed to create a Japanese joking conversational system. PUNDA Simple is based on a simplified version of the algorithm of the main PUNDA system, which, although still under development, at its current state can be used as a pun generating support tool.

Although PUNDA Simple was created for the need of this research, the main part of the algorithm is similar to the one used in the main system.

The algorithm. The PUNDA Simple algorithm consists of two parts: Candidate Selection Algorithm (CaSA) and Sentence Integration Engine (SIE) – see Figure 1.

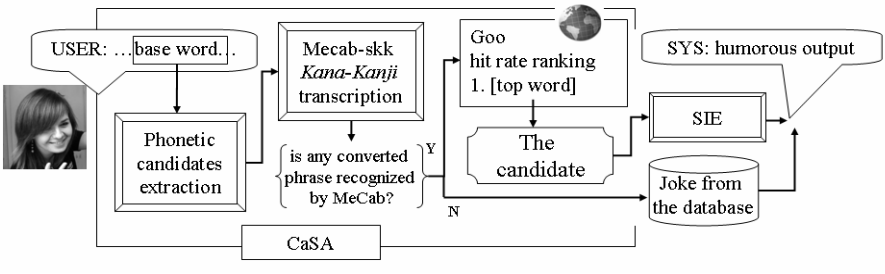


Fig. 1. Algorithm outline for PUNDA Simple joke generating engine

CaSA. In this step, the system generates a candidate for a pun. The input is a sentence, from which a base word for pun (a word that will be transformed into a pun) is selected. The input is analyzed by morphological analyzer MeCab [9], and if any element is recognized as an ordinary noun, it becomes the base word for the pun (a preliminary experiment proved that most of *dajare* base words are ordinary nouns). If no ordinary noun is found, one medium-sized (with medium amount of characters) sequence of characters recognized by MeCab as an independent word is selected randomly. Then, for the base word, pun candidates are generated using 4 generation patterns: homophony, initial mora addition, internal mora addition and final mora addition. For example, for the word *katana* (a Japanese sword), the process goes as follows:

base word: {katana}
candidates:
1. **homophony**: {katana}
2. **initial mora addition**: {*katana } (akatana, ikatana, ukatana...)

3. **final mora addition**: { katana*} (katanaa, katanai, katanau...)
4. **internal mora addition**: {ka*tana}, {kata*na} (kaatana, kaitana, kautana...)
(with * meaning one single mora).

The candidates are generated in *Kana* characters (one character = one syllable). In the next step, for each candidate a list of possible *Kanji* (Japanese ideograms) transcription is extracted using MeCab-skkserv *Kana-Kanji* Converter [10]. Then, if any of converted sequence of characters was recognized by the morphological analyzer as an existing word, its hit rate was checked in the Internet. The candidate with highest Goo hit rate was extracted as a pun candidate. For example, for the word *katana* the candidate would be *takatana* (a high shelf).

SIE. In this step, a sentence including the candidate extracted by CaSA is generated. To make system's response more related to the user's input, each sentence that included joke started with the pattern "[base word] *to ieba*" ("Speaking of [base word]"). The following part of the sentence is extracted from KWIC on WEB online Keyword-incontext sentences database [11] - the candidate is used as a query word and the list of sentences including this word is extracted. All sentences that contain non-Japanese characters (such as Latin letters, brackets, numbers etc.) are deleted, as they may be difficult to understand without the context. Then, the shortest sentence with exclamation marks is selected (we assumed that most jokes convey some emotions, and exclamation marks are its visible sign). If no such sentence is found, the shortest sentence that ended with a period is selected. Finally, the part following the candidate is extracted and added as the second part to the system's response – see the example below:

User: - *Natsu wa atsui desu kedo, Toukyou hodo ja nai desu*.(The summer is hot, but not as hot as in Tokyo) [base word: *natsu* (summer), pun candidate: *natsukashii* (emotionally attached; loved)]

System: -*Natsu to ieba, natsukashii Nose de, kyuuyuu to waiwai!* (Speaking of summer, it will be fun to meet some old friends in my beloved Nose! <a town near Osaka>)

If no appropriate sentence is found on KWIC, the response ends with "[candidate] *da yo ne*" ("it's the [candidate], isn't it") – see the example below.

User: -*Ame da kara kaimono iku no wa mendokusai ne.* (It's raining so going shopping is a bad idea) [base word: *ame* (rain), pun candidate: *ame* (a candy)]

System: -*Ame to ieba ame da yo ne*. (Rain reminds me of candies.)

When the candidates list is empty, the system selects one random pun from Sjöbergh's pun database [12] – see the example below.

User: - *Sou, atarashii baito ne.* (Yeah, it's my new part-time job) [base word: *baito* (a part-time job), pun candidate: none]

System's response (from the pun corpus): -*Kureyon wo katte kureyon!* (Buy the pen, man!).

2.3 Pundalin

The two systems described above were merged to create a joking conversational system Pundalin. The rule was very simple – in every third dialogue turn Modalin's output was replaced by PUNDA Simple's joke. In other words, every third user's utterance became PUNDA's input and an appropriate pun for it was generated, using the algorithm described in section 2.2. This method, albeit quite simple, allowed us to check if the usage of humor improved the system's overall quality.

3 Two Evaluation Experiments

To check if humor can enhance the non-task oriented dialogue, we conducted two evaluation experiments, using Modalin as the baseline system and Pundalin as the main, humor-equipped system.

3.1 User's Evaluation

In the first experiment, users were asked to perform a 10-turn dialogue with Modalin, and then with Pundalin. No topic restrictions were made, so that the talk could be as free and human-like as possible. Thus, the utterances variety was quite big – most of them, however, resembled normal (human-like) beginning of conversation, for example: "What did you do yesterday?", "May I ask you a question?" or "It's hot today, isn't it?"

There were 13 participants of the experiment, 11 males and 2 females; all of them were university undergraduate students. After talking with both systems, they were asked to fill a questionnaire about each system's performance. The questions were: A) Do you want to continue the dialogue?; B) Was the system's talk grammatically natural?; C) Was the system's talk semantically natural?; D) Was the system's vocabulary rich?; E) Did you get an impression that the system possesses any knowledge?; F) Did you get an impression that the system was human-like?; G) Do you think the system tried to make the dialogue more funny and interesting? and H) Did you find system's talk interesting and funny?. The answers for questions were given in 5-point scale with some explanations added. Option "I don't know" was also available, however none user did chose it. Each user filled two such questionnaires, one for each system. The final, summarizing question was: "Which system do you think was better?". Although the question may seem too general, it was added on the basis of common sense. Also, thanks to this last question we do not have to rely on detailed questions only, which gives us another interesting point of view in the evaluation experiment.

3.2 Third Person Evaluation

To verify user's assessment, a third person evaluation experiment was conducted. For each Modalin (non-humorous) dialogue from user's experiment, first three turns were selected, and user's third utterance was manually used as an input for PUNDA Simple to generate a humorous response. The reason for shortening the dialogues was that we wanted to check also the "Modalin plus PUNDA" version, for it was much easier to compare it with Modalin-only dialogue (the content was the same, apart from the last

system's response). Therefore, after altering the third system's response, the rest of the dialogue became irrelevant, and for this experiment's purpose it had to be ended after this turn.

Each participant evaluated 1 set, including 3 short dialogues: Modalin only, Modalin plus PUNDA (with the third system's response replaced by PUNDA's joke) and Pundalin. There were 13 sets of dialogues, and each of them was evaluated 5 times. For the dialogues were too short to be evaluated in details, in the questionnaire one general question was: "Which dialogue do you find most interesting and funny?"

The evaluators were 65 university students, 37 male and 28 female. They were not told of the origin of dialogues and apparently did not know that some of evaluated utterances were generated by computer.

4 Results

The results of these two experiments clearly show, that humor enhances non-task oriented dialogue, as in all categories humor-equipped systems received higher scores and were evaluated as better and more interesting than non-humor systems.

4.1 User's Evaluation

11 out of 13 users (84.6%) evaluated Pundalin (humor-equipped system) as better than Modalin (non-humor - equipped) – see Figure 2. This shows that implementation of humor enhanced the conversational system's performance.

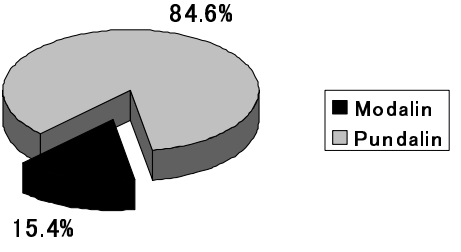


Fig. 2. User's evaluation – results for the question "Which system do you think was better?"

Pundalin received higher scores also in detailed questions (see Table 1). The difference was especially visible in questions G and H, which we consider to be of high relevance in humor research, as they directly address the presence and positive role of humor in the dialogues.

The results of detailed questions evaluation were checked for significance using student t-test (paired). Apart from questions A and B (P value > 0.05), all the results turned out to be statistically significant on 5% level. However, overall results of both systems were relatively lower than they could be. Possible reasons for these are discussed in section 5.

Table 1. User's evaluation – results for Modalin (non-humor equipped system) and Pundalin (humor-equipped system) for detailed questions: A) Do you want to continue the dialogue?; B)Was the system's talk grammatically natural?; C) Was the system's talk semantically natural?; D) Was the system's vocabulary rich?; E) Did you get an impression that the system possesses any knowledge?; F) Did you get an impression that the system was human-like?; G) Do you think the system tried to make the dialogue more funny and interesting? and H) Did you find the system's talk interesting and funny? Answers were given in a 5-point scale.

Questions	A	B	C	D	E	F	G	H
Modalin	2.62	2.15	1.85	2.08	2.15	2.38	1.92	2.46
Pundalin	3.38	2.92	2.69	3.00	2.85	3.31	4.15	4.08
difference	0.76	0.77	0.84	0.92	0.70	0.93	2.23	1.62

4.2 Third Person Evaluation

Also in the second experiment, humor-equipped system scored higher than the non-humorous one. The question asked in this evaluation was: "Which dialogue do you find most interesting and funny?" Evaluators could choose between 3 options: Dialogue 1 (Modalin's first 3 turns), Dialogue 2 (Modalin's first 3 turns with system's third response replaced by PUNDA) and Dialogue 3 (Pundalin's first 3 turns).

Among 65 evaluators, only 10 (15.4%) responded that Dialogue 1 was most interesting and funny. 20 (30.8%) pointed out Dialogue 2 and 35 – Dialogue 3 (53.8%). This means that each of humor containing dialogues received evaluation clearly higher than non-humor dialogue (84.6% as a sum of both humorous dialogues) - see Figure 3.

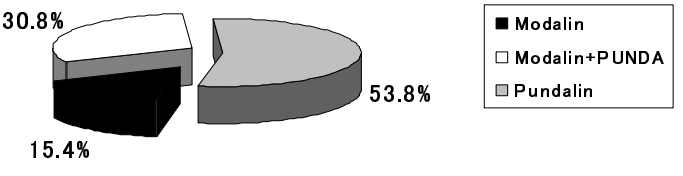


Fig. 3. Third person evaluation - results for the question "Which dialogue do you find most interesting and funny?" (short dialogues)

Although it is clear that both humorous dialogues received much better score, such high score for Pundalin was rather surprising. This is discussed in section **5**.

5 Discussion

Our presumptions turned out to be correct – humor did improve the talking system's performance. In both experiments, Pundalin's dialogues received higher scores than Modalin, in the general evaluation as well as in detailed questionnaire.

As far as the second one is concerned, the most important and relevant for this research are questions G (Do you think the system tried to make the dialogue more

funny and interesting?) and H (Did you find the system's talk interesting and funny?). At these two categories, the differences between the two systems are clearly visible (2.23 point and 1.62 point respectively). This means, that not only did the system try to amuse the user, but the attempts were generally successful.

Also the increase in other categories, such as the will of continuing the dialogue (question A), the possession of knowledge (question E) or system's human-likeness (question F), albeit not directly related to humor, were probably affected by its presence. This also proves that the implementation of joking engine can make talking systems more natural and close to humans.

Remaining categories (questions B, C and D) are all related to system's language abilities. Although it may seem that the presence of humor can distract user's attention from grammatical mistakes or semantic irrelevance, the reason for Pundalin's victory in these aspects probably is that it uses human-created sentences, which obviously are more correct than these generated by Modalin.

Another interesting issue in the experiment results was the the fact that Pundalin scored 23% higher than Modalin+PUNDA in the third person evaluation. The reason for providing the experiment participants also with Modalin+PUNDA dialogue was that we thought it should be easier to tell the difference when comparing it to the Modalin only dialogue (which, apart from the last response, was exactly the same as Modalin+PUNDA). However, the results proved that it does not necessarily have to be true, as more users pointed at Pundalin as the funnier and more interesting system. One possible explanation is that that Pundalin's jokes were funny enough to drag the evaluators' attention and convinced at least some of them that this system was better than two others.

6 Additional Experiment

The results described above encouraged us to convey another experiment, in which third person evaluators would assess the whole-length Modalin and Pundalin dialogues. For few reasons, we expected the results to be slightly lower than the previous ones. First, the joking engine has no context integration algorithm, which is obviously more visible in the long dialogues. Also, reading jokes-including conversations should be less funny than getting a humorous answer from the partner in a conversation (comparing to the users' evaluation).

6.1 The Method

The questionnaires for long dialogues were similar to these used in user's evaluation experiment, with few significant differences. First, the word "system" was changed to "dialogue" or "speaker", so that the participants would not know that some utterances were computer-generated (in the chat logs speakers were marked as "Speaker A" for the user and "Speaker B" for the system). For the same reason, the question F (about human-likeness) was deleted. Also, in questions B, C, D, E and G two sub-sections were added: 1) "Speaker A" and 2) "Speaker B" – so that the dialogue participants

could be evaluated separately. For question H, these two options plus a third one: "overall" – were added. In the end, evaluators had to answer the final question, the same as in the previous experiment - "Which dialogue do you find most interesting and funny?"

The chat logs were divided into 13 sets, each of which included one Modalin and one Pundalin dialogue. Every set was evaluated by 5 participants, what makes a total of 65 evaluators. All of them were university students.

6.2 Results

As we expected, the results for long dialogues were not that good as in previous experiments. However, the overall question's results still show that humorous system is visibly better than the non-humorous one. 45 out of 65 evaluators (69.2%) pointed at Dialogue 2 (Pundalin) as more interesting and funny (see Figure 4).

For the remaining questions, we only take in consideration results for Speaker B (as Speaker A was human, which we do not evaluate here). In all questions, Pundalin dialogues received higher scores than Modalin (see Table 2).

Table 2. Third person evaluation – results for Modalin (non-humor equipped system) and Pundalin (humor-equipped system) for detailed questions: A) Do you want to read the continuation of the dialogue?; B)Was Speaker B's talk grammatically natural?; C) Was the Speaker B's talk semantically natural?; D) Was the Speaker B's vocabulary rich?; E) Did you get an impression that the Speaker B possesses any knowledge?; F) <deleted – see section **6.1**>; G) Do you think the Speaker B tried to make the dialogue more funny and interesting?; H-1) Did you find the dialogue interesting and funny in general? and H-2) Did you find Speaker B's talk interesting and funny? Answers were given in a 5-point scale.

Questions	A	B	C	D	E	F	G	H-1	H-2
Modalin	2.60	1.78	1.48	2.03	1.87	X	2.51	2.88	2.73
Pundalin	2.89	2.09	1.69	2.38	2.13	X	2.91	3.19	3.16
difference	0.29	0.31	0.21	0.35	0.26	X	0.40	0.31	0.43

Although all results showed that Pundalin was better, the student t-test (unpaired[1]) showed that – apart from questions D and G – the results are not statistically significant on 5% level. This issue is discussed in section **6.3**.

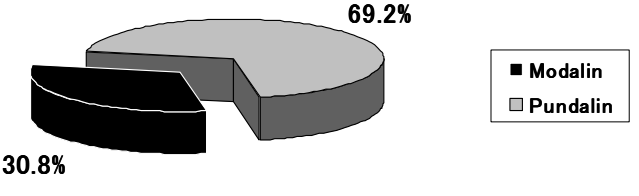


Fig. 4. Third person evaluation - results for the question "Which dialogue do you find most interesting and funny?" (long dialogues)

[1] The reason we decided to conduct unpaired test was that some evaluators choose the "I don't know" answer (also available on the form). Such answers were not taken into consideration when counting points, and this caused differences between numbers of valid answers for each question.

6.3 Discussion

Consistently with previous results, also the third person long dialogues evaluation experiment showed that humor-equipped system is generally better. As we expected, the differences between the two systems were slightly smaller and most of the results are considered not statistically significant (P value > 0.05). However, in one of the most relevant to our research question G ("Do you think the Speaker B tried to make the dialogue more funny and interesting?"), the results were significant (P < 0.05) – this clearly shows, that also third person evaluators at least appreciated Speaker B's (system's) "effort" to make the talk more interesting.

Although most of the results were not statistically significant, in the overall third person evaluation Pundalin still scores much higher than Modalin (69.2% vs. 30.8%). This shows that – even when the differences between dialogues are not statistically significant – the evaluators preferred the humorous ones. Obviously, Pundalin's rich vocabulary as well as grammatical and semantic correctness also had an influence on evaluator's preferences, however, there is a high possibility, that humor is also a factor that "makes things better" here. This, however, is hard to confirm empirically.

We suppose that generally smaller (and less significant) differences in this experiments were caused by the lack of context integration algorithm in pun generator – for a third (non-user) person, jokes generated by the system may seem irrelevant to the topic. Although users probably get the same impression, the very fact that system responded to their utterance with a joke apparently triggered them to evaluate it higher.

This leads us to another interesting point of view on the subject – does the knowledge that the partner (in user's evaluation) or one of the speakers (in third person evaluation) is a computer system influence the results? In case when evaluators know that, do they admire the system for trying to tell jokes, or rather dislike it for doing it less swiftly than humans? The results of user's evaluation (question F) showed, that "humanlikeness" of the system increased - however, this issue still requires more investigation.

In this study, in the third person evaluation experiment, the participants were not told of the origin of the dialogues. Therefore, there is a need of conducting a similar experiment, in which the participants would know that the dialogues took place between humans and computer systems.

7 What Solutions Do We Propose?

The overall message coming from this experiment is that research on humor processing is a worthwhile enterprise and is definitely worth continuing. However, of course, many improvements are still to be made, in talking system itself as well as in pun generator. Also, new evaluation methods will have to be considered, as assessing the "quality" of humor is difficult even in case of humans. Some ideas on these subjects are discussed below.

7.1 Solution to the Timing Problem

One of important problems in humor processing is the timing of jokes. Obviously, it is not always appropriate to tell a pun – for example, when the user is angry, a badly

timed joke may only boost his anger. One solution to the timing problem would be an implementation of an emotive analysis system, that would detect the user's emotional states and decide, whether it is proper to tell a pun. At this point of research, we are planning to use Ptaszynski's MLAsk Emotive Analysis System [13] and combine it with PUNDA's algorithm.

7.2 Evaluation Methods

Evaluation of joking systems is a serious problem in the field of humor processing. Even if we construct a perfect set of most subject-relevant questions, after all, it is humor that we assess – and this, by definition, can not be completely objective. However, some methods may prove useful as far as joking conversational systems are concerned. We are considering conducting of humor-oriented Turing Test [14], in which evaluators will have no knowledge of who their conversation partner is, and afterwards will have to guess, if it was a human or a machine. The experiment would include a comparison between humorous and non-humorous systems. This method, albeit not perfect, will presumably give credible results for the humanlikeness category of system's evaluation.

8 Conclusion

In this paper, we proved that implementation of a joke generator into a non-task oriented conversational system enhances its performance. Humor-equipped system was evaluated higher than the non-humor one, by the users as well as third person evaluators. We discussed the results, pointed out some problems and proposed their possible solutions. However, what is more important, the overall results of these experiments show, that combining the field of humor processing with research project on freely talking systems is a step in the right direction and under no circumstances should not be abandoned in the future.

Acknowledgments. This work was partially supported by the Research Grant from Nissan Science Foundation.

References

1. Morkes, J., Kernal, H.K., Nass, C.: Effects of humor in task-oriented human-computer interaction and computer-mediated communication: A direct test of srct theory. Hum-Comput. Interact. 14(4), 395–435 (1999)
2. Reeves, B., Nass, C.: The media equation: How people treat computers, television, and new media like real people and places. Cambridge Univ. Press, Cambridge (1996)
3. Binsted, K.: Using humour to make natural language interfaces more friendly. In: Proceedings of the AI, ALife and Entertainment Workshop, Intern. Joint Conf. on Artificial Intelligence (1995)
4. Binsted, K.: Machine humour: An implemented model of puns. Univ. of Edinburgh (1996)

5. Loehr, D.: An integration of a pun generator with a natural language robot. In: Hulstijn, J., Nijholt, A. (eds.) Proc. Intern. Workshop on Computational Humor, pp. 161–172. University of Twente, Netherlands (1996)
6. Dybala, P., Ptaszynski, M., Rzepka, R., Araki, K.: Extracting Dajare Candidates from the Web - Japanese Puns Generating System as a Part of Humor Processing Research. In: Proceedings of LIBM 2008, pp. 46–51 (2008)
7. Higuchi, S., Rzepka, R., Araki, K.: A casual Conversation System Using Modality and Word Associations Retrieved from the Web. In: Proceedings of EMNLP, Waikiki, Hawaii (to appear, 2008)
8. Goo search engine, http://www.goo.ne.jp/
9. Kudo, T.: MeCab: Yet another part-of-speech and morphological analyzer (2001), http://mecab.sourceforge.ne
10. MeCab-skkserv Kanji-Kana converter, http://chasen.org/~taku/software/mecab-skkserv/
11. Yoshihira, K., Takeda, T., Sekine, S.: KWIC system for Web Documents (in Japanese). In: Proceedings of the 10th Annual Meetings of the Japanese Association for NLP, pp. 137–139 (2004)
12. Sjöbergh, J., Araki, K.: Robots Make Things Funnier. In: Proceedings of LIBM 2008, pp. 46–51 (2008)
13. Ptaszynski, M., Dybala, P., Rzepka, R., Araki, K.: Effective Analysis of Emotiveness in Utterances Based on Features of Lexical and non-Lexical Layer of Speech. In: Proceedings of NLP Conference, Tokyo (2008)
14. Turing, A.: Computing Machinery and Intelligence. Mind 59(236), 433–460 (1950)

Improving Transductive Support Vector Machine by Ensembling*

Tao Li and Yang Zhang

College of Information Engineering,
Northwest A&F University,
Yangling, Shaanxi Province,
P.R. China, 712100
1648708531t@nwsuaf.edu.cn
zhangyang@nwsuaf.edu.cn

Abstract. Transductive Support Vector Machine (TSVM) is a method for semi-supervised learning. In order to further improve the classification accuracy and robustness of TSVM, in this paper, we make use of self-training technique to ensemble TSVMs, and classify testing samples by majority voting. The experiment results on 6 UCI datasets show that the classification accuracy and robustness of TSVM could be improved by our approach.

1 Introduction

Currently, semi-supervised learning is a hot research problem. Generally speaking, it is rather expensive to obtain labeled training samples, while unlabeled data are easy to get. Hence, it is much helpful if we can learn from labeled data and unlabeled data all together. Transductive Support Vector Machine (TSVM) is an extension of standard support vector machines with unlabeled data. In a standard SVM only the labeled data is used, and the goal is to find a maximum margin linear boundary in the Reproducing Kernel Hilbert Space. In a TSVM the unlabeled data is also used. The goal is to find a labeling of the unlabeled data, so that a linear boundary has the maximum margin on both the original labeled data and the (now labeled) unlabeled data [1]. Vikas Sindhwani et al. [2]propose a variant of TSVM that involves multiple switching of labels. During the trial, we found it has very good results in some data sets, but in other data sets or the data sets which have fewer training dataset perform badly.

Ensemble learning is proved to be very successful in practice by data mining research community. In order to further improve the classification performance of TSVM, in this paper, we propose En-TSVM, which ensembles TSVM with help of self-train technique, and classifies testing samples by majority voting. The experiment result shows that En-TSVM has better classification accuracy and robustness than TSVM.

* This work is supported by Talent Fund of Northwest A&F University (01140402) and Young Cadreman Supporting Program of Northwest A&F University (01140301). Corresponding author: Yang Zhang.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 226–235, 2008.

The paper is organized as follows. Section 2 reviews the related work. The algorithm of En-TSVM is described in section 3. The detailed experiment setting and results are shown in section 4, followed by our conclusion and future work in section 5.

2 Related Work

Currently, there is a rapid improvement on TSVM in the research community of semi-supervised learning. Ulf Brefeld et al. [3] developed the principle of maximizing the consensus among multiple independent hypotheses into a semi-supervised support vector learning algorithm for joint input output spaces and arbitrary loss functions. Olivier Chapelle et al. propose to use a global optimization technique known as continuation to alleviate the main problem that the optimization problem is non-convex and has many local minima, which often results in suboptimal performances [4]. Yihong Liu et al. [5] proposed an improved algorithm for text classification named double transductive inference algorithm based on TSVM.

Literature [1] is a more comprehensive overview of semi-supervised learning. It also gives a simple introduction to TSVM and the self-training technique. By taking advantages of the correlations between the views using canonical component analysis, Zhi-Hua Zhou et al. proposes a method working under a two-view setting, which perform semi-supervised learning with only one labeled training example [6].

It is concluded in many research that the ensemble approach helps to improve the effect of learning [7,8]. Dietterich et al. have proved that Multiple Classifiers System (MCS) can improve classification performance in many applications [7]. Zhenchun Lei et al. propose to ensemble SVMs for text-independent speaker recognition, and the bagging-like model and boosting-like model are proposed by adopting the ensemble idea [9]. Hsuan-Tien Lin and Ling Li formulate an infinite ensemble learning framework based on SVM [10]. Yan-Shi Dong et al. proposed two types of ensembles on SVM classifiers, the data partitioning ensembles and heterogeneous ensembles, and experimentally evaluated them on three well-accepted datasets [11]. Major conclusions are that disjunct partitioning ensembles with stacking could achieve the best performance, and that the parameter varying ensembles are proven to be effective, meanwhile have the advantage of being deterministic.

Recently, there is some research on ensemble for semi-supervised learning. Erdisci et al. used an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems [12]. Zhi-Hua Zhou et al. propose tri-training [13], a new co-training style semi-supervised learning algorithm. Tri-training is a special self-training, with the final classification result of the ensemble being determined by majority voting.

In this paper, we make use of self-training technique to train an ensemble of TSVM, and classify testing samples by majority voting.

3 Ensemble TSVM

3.1 A Brief Introduction to TSVM

The research on SVM based Transductive learning algorithm is still in its initial stage, with TSVM being the most important research result. Unlike SVMs, the formulas of TSVMs lead to a non-convex optimization problem [14]. Here, we give a simple introduction to TSVM.

Giving a set of labeled training samples, which follow the same data distribution, and are independent to each other:

$$(x_1, y_1), \ldots, (x_n, y_n), \tag{1}$$

$x_i \in R^m$, $y \in \{-1, +1\}$, and another set of unlabeled training samples, which follow the same data distribution:

$$x_1^*, x_2^*, x_3^*, \ldots, x_k^*, \tag{2}$$

the following optimization problem is setup for standard TSVM [15]:

$$\begin{aligned} Minimize \quad over \ (\ & y_1^*, \ldots, y_k^*, w, b, \xi_1, \ldots, \xi_n, \xi_1^*, \ldots, \xi_n^*) \\ & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + C^*\sum_{j=1}^{k}\xi_j^* \\ s.t. : \forall_{i=1}^{n} \ & : \ y_i[w \cdot x_i + b] \geq 1 - \xi_i \\ \forall_{j=1}^{k} \ & : \ y_j[w \cdot x_j^* + b] \geq 1 - \xi_j^* \\ \forall_{i=1}^{n} \ & : \ \xi_i \geq 0 \\ \forall_{j=1}^{k} \ & : \ \xi_j^* \geq 0 \end{aligned} \tag{3}$$

Here, parameter C^* represents the impact factor of unlabeled training samples in the training process, and $C^*\xi_j^*$ represents the impact factor of unlabeled training sample x_j^* int the objective function.

In supervised learning, the discriminant function of SVM is:

$$f(x) = w_0 \cdot x + b_0. \tag{4}$$

In semi-supervised learning, TSVM constructs the classification hyper-plane by inductive learning on labeled training samples, and get the discriminant function value for each unlabeled samples following formula (4). By using current labeled samples for study summarized, TSVM gets the current split-plane and the discriminate function as (4), and calculates all current discriminant function values of the unlabeled samples. Following formula (5) and (6), two unlabeled samples with largest, and least discriminant function value could be selected out from the boundary area of current classification hyper-plane, respectively.

$$\max(f(x^*)), s.t. 0 < (f(x_i^*)) < 1 \tag{5}$$

$$\min(f(x^*)), s.t. -1 < (f(x_i^*)) < 0 \tag{6}$$

3.2 Ensemble TSVM

In this paper, we ensemble a set of TSVM classifiers for semi-supervised learning, and classify unknown samples by majority voting. Here, the TSVM classifiers are trained by self-training approach.

(1) Training algorithm

Suppose $L = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{|L|}, y_{|L|})\}$ is the set of labeled training samples, and $U = \{x_1^*, x_2^*, \ldots, x_{|U|}^*\}$ is the set of unlabeled samples, then, $U \cup L$ is the set of training samples. Initially, we assign the weight of labeled samples and the unlabeled samples to $\frac{1}{|L|}$ and $\frac{1}{|U|}$, respectively. And then, e_1 is trained on $U \cup L$ with help of W. A pseudo-label is set to samples in $\{x^*|x^* \in U, |e_1(x^*)| > Tr\}$, and samples with pseudo-label are moved from U to L. Hence, we get a new set of labeled samples L, and a new set of unlabeled samples U. The same process is repeated on $U \cup L$ for T times, and we could get an ensemble of T classifiers $\{e_1, e_2, \ldots, e_{|T|}\}$.

Algorithm 1. Training Algorithm for En-TSVM

Input:

The training samples, $L \cup U$;
The number of loops, T;
Threshold, T_r.

Output:

The ensemble of TSVM, E.

$E = \phi$;
for $m = 1$ To T **do**
  for each $x_i \in L$ **do**
    $W_{x_i} = \frac{1}{|L|}$;
  end for
  for each $x_i \in U$ **do**
    $W_{x_i} = \frac{1}{|U|}$;
  end for
  $e_m = TSVM(L, U, W)$;
  $E = E \cup \{e_m\}$;
  for each $x_i \in U$ **do**
    if $e_m(x_i) > T_r$ **then**
      $L = L \cup \{(x_i, 1)\}, U = U - \{x_i\}$;
    end if
    if $e_m(x_i) < -T_r$ **then**
      $L = L \cup \{(x_i, -1)\}, U = U - \{x_i\}$;
    end if
  end for
end for
return E

(2) Classification algorithm

Majority voting is one of the simplest and most intuitive ways for ensembling base classifiers [16]. In this paper, we take majority voting as our strategy

for classifying testing samples by the classifier ensemble. Let t be the testing sample, $vote_{pos}$ be the number of base classifiers in E which classify t into positive class, and $vote_{neg}$ be the number of base classifiers in E which classify t into negative class, the class label predicted by E could be determined by $max(vote_{pos}, vote_{neg})$. The classification algorithm of En-TSVM is illustrated in Algorithm 2.

Algorithm 2. Classification Algorithm for En-TSVM

Input:
The ensemble of TSVM, E;
The testing sample, t.
Output:
the class label.

```
vote_pos = 0; vote_neg = 0;
for each e_i ∈ E do
   if e_i.classify(t) == +1 then
      vote_pos ++;
   end if
   if e_i.classify(t) == −1 then
      vote_neg ++;
   end if
end for
if vote_pos ≥ vote_neg then
   return +1;
else
   return −1;
end if
```

4 Experiments Results

In order to test the classification performance of proposed approach, we made experiments on 6 *UCI* datasets[1]. We implemented our algorithm in C with help of SVMLIN[2] software. Our experiments were made on a PC with Core 2 CPU, Windows XP, and 1GB memory.

In our experiments, we set $T = 7$, and $T_r = 0.9$. We conducted 4 groups of experiment. For each experiment setting, the averaged classification performance of 200 experimental trails are reported here as the final result.

4.1 Experiment A

In this group of experiment, we randomly cut the original dataset into dataset A and dataset B, with $|A| \approx 2|B|$ or $|A| \approx |B|$. We randomly select r percent of samples from dataset A for training, and test on the dataset B. We compare

[1] URL: http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/
[2] URL: http://people.cs.uchicago.edu/∼vikass/svmlin.html

Table 1. The result of the experiment A

Dataset	#Att	#L	#U	#T	r%	En-TSVM		TSVM		t-Test
						Acc%	MSD	Acc%	MSD	
adult	123	800	31761	16281	70	**82.4**	**0.039**	81.3	0.052	+
					80	**82.7**	**0.084**	81.7	0.136	+
					90	**82.9**	**0.095**	81.9	0.103	+
australian credit approval	14	120	320	250	70	**82.7**	0.091	82.1	**0.068**	+
					80	**82.8**	0.058	81.9	**0.018**	+
					90	**83.2**	0.047	82.2	**0.039**	+
breast cancer wisconsin	10	100	377	206	70	**96.6**	**0.063**	96.1	0.071	+
					80	**98.3**	**0.044**	97.4	0.086	+
					90	**99.1**	**0.013**	98.6	0.039	+
mushrooms	112	400	5500	2224	70	**98**	**0.006**	96.6	0.097	+
					80	**97.3**	**0.005**	97.1	0.026	+
					90	**97.4**	**0.002**	96.7	0.018	+
german credit	24	120	380	500	70	**69.2**	**0.045**	68.7	0.072	+
					80	**69.3**	0.061	68.7	**0.056**	+
					90	**70.4**	**0.036**	69.4	0.043	+
ionosphere	34	80	119	152	70	**79.4**	**0.017**	78.8	0.021	+
					80	**80.1**	**0.008**	79.6	0.016	+
					90	**82.4**	0.077	82	**0.069**	+

the classification performance of En-TSVM with TSVM, and the classification performance is measured by accuracy and Mean Square Deviation (MSD). Furthermore, t-Test is checked to examine whether the improvement in classification accuracy of En-TSVM with respect to TSVM is significant or not. The experiment result is shown in table 1.

In table 1, column 1 lists the name of the datasets used; column 2, column 3, column 4, and column 5 gives the number of the attribute, the number of labeled sample in A, the number of unlabeled sample in A, and the number of samples in B, respectively; column 6 gives the percentage r; column 7 and 8 presents the classification performance of En-TSVM and TSVM, respectively; the last column shows the results of t-Test, with +(-) means the result of t-Test is significant (not significant).

From Table 1, it is obvious that our En-TSVM has better classification accuracy and classification robustness than single TSVM.

4.2 Experiment B

In this group of experiment, we cut the dataset into two parts in the same way as in experiment A. We randomly select $|L|$ labeled samples, and all the unlabeled samples from dataset A for training, a testing on dataset B. The experiment result is shown in Fig. 1.

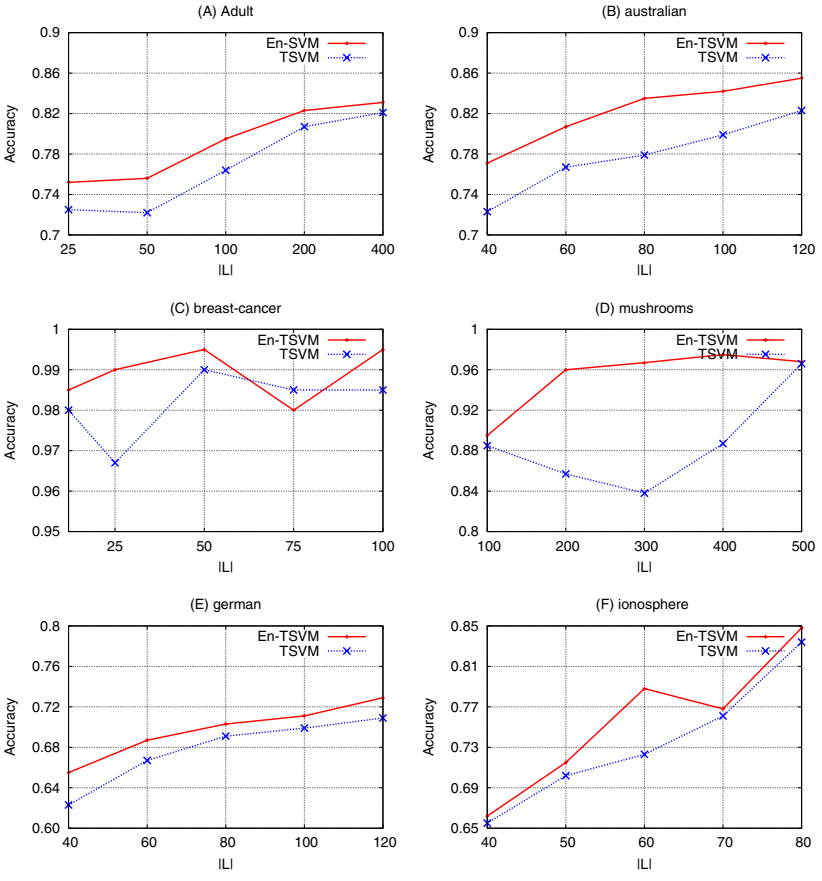


Fig. 1. The result of the experiment B

In each sub-figure of Fig. 1, the horizontal axis represents $|L|$, and the vertical axis represents the classification accuracy. From Figure 1, it is obviously that for most of the cases the classification accuracy of En-TSVM is better than TSVM.

4.3 Experiment C

In this group test, we make experiment in the hold-out way, with $3/4, 2/3, 1/2, 1/3$, and $1/4$ part of the original dataset being used as training dataset, the rest of the samples being used as testing dataset, respectively. The number of labeled samples in training dataset is set in the same way as in experiment A. Here, we consider that whether the ensemble TSVM is a better classifier or not with decreasing size of training dataset. The experiment result is shown in table 2.

In table 2, column 1 lists the name of the datasets used; column 2 gives the number of labeled samples in the training dataset; column 3 gives the name of

Table 2. The result of the experiment C

Dataset	#L	Classifier	Accuracy%				
			3/4	2/3	1/2	1/3	1/4
adult	800	En-TSVM	**83.23**	**83.6**	**83.17**	**82.95**	**83.49**
		TSVM	82.84	82.81	82.68	82.88	82.92
australian credit approval	120	En-TSVM	**83.63**	**83.41**	**84.59**	**83.66**	**83.17**
		TSVM	82.46	77.73	82.85	82.57	82.4
breast cancer wisconsin	100	En-TSVM	**96.61**	**96.2**	**94.54**	**94.01**	**94.01**
		TSVM	94.92	94.94	93.7	93.56	93.56
mushrooms	400	En-TSVM	**98.28**	**81.2**	**85.79**	**55.77**	**58.08**
		TSVM	96.8	80.46	85.43	54.18	55.47
german credit	120	En-TSVM	**72.69**	**69.58**	**72.95**	**70.94**	**67.73**
		TSVM	70.28	68.98	70.94	68.68	67.13
ionosphere	80	En-TSVM	**96.55**	**94.83**	**94.29**	**94.29**	**83.59**
		TSVM	94.25	93.1	88.57	88.57	81.68

the classifier; column 4 presents the classification accuracy with different ratio of training dataset with respect to the original dataset.

It is obvious that En-TSVM outperforms TSVM at various ratios.

4.4 Experiment D

In this group of experiment, we measure time complexity of our algorithm. Here, we only report our experiment result on *mushrooms* dataset for lacking of space. The similar experiment result could be observed on other datasets listed in table 1.

In experiment D1, we set $|U \cup L| = 5900$, and we change the number of labeled samples $|L|$ in training sets. In experiment D2, we set $|L| = 400$, and increase

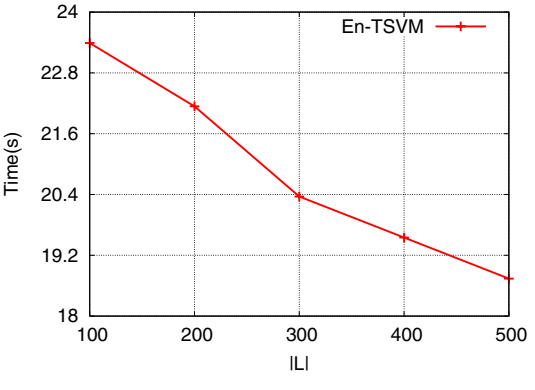


Fig. 2. The result of experiment D1

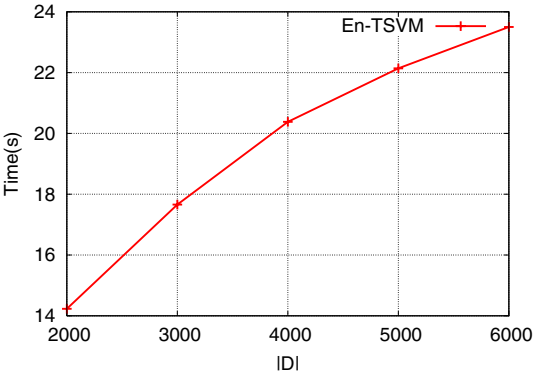


Fig. 3. The result of experiment D2

the number of training samples $|U \cup L|$. The result of experiment D1 and D2 are shown in Fig. 2 and Fig. 3, respectively.

The horizontal axis in Fig. 2 represents the size of labeled training dataset $|L|$, and the horizontal axis in Fig. 3 represents the size of training dataset $|U \cup L|$,here,we set $|D|=|U \cup L|$. The vertical axis of both Fig. 2 and Fig. 3 represents the time needed for training (in second).

From Fig. 2 and Fig. 3, it could be observed that the running time needed for training decrease linearly with respect to the increasing of $|L|$, and increases linearly with respect to the increasing of $|U \cup L|$.

5 Conclusions and Future Work

On the basis of TSVM, in this paper, we propose En-TSVM, which ensembles TSVM with help of self-train technique, and classifies testing samples by majority voting. Experiment result shows that compared with TSVM, our En-TSVM has classification accuracy and classification robustness.

In the future, we schedule to study other approaches for combining TSVM to further improve the classification performance of TSVM.

References

1. Zhu, X.: Semi-supervised learning literature survey. Computer Science TR (1530), University of Wisconsin - Madison (February 2006)
2. Sindvani, V., Sathiya Keerthi, S.: Large Scale Semi-Supervised Linear SVMs. In: SIGIR 2006, Agust 6-11 (2006)
3. Brefeld, U., Scheffer, T.: Semi-Supervised Learning for Structured Output Variables. In: Proceedings of the 23rd international conference on Machine Learning, Pittshurgh, PA (to appear, 2006)

4. Chapelle, O., Chi, M., Zien, A.: A continuation method for semisupervised SVMs. In: ICML 2006, 23rd International Conference on Machine Learning, Pittsburgh, USA (2006)
5. Liu, Y., Teng, G., Yang, J.M., Wang, F.: Double Transductive Inference Algorithm For Text Classification. In: ICIC International, December 2007, pp. 1463–1469 (2007)
6. Zhou, Z.-H., Zhan, D.-C., Yang, Q.: Semi-supervised learning with very few labeled training examples. In: Twenty-Second AAAI Conference on Artificial Intelligence, AAAI 2007 (2007)
7. Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, p. 1. Springer, Heidelberg (2000)
8. Polokar, R.: Ensemble based systems in decision making. IEEE Circuits and System Magazine (2006)
9. Lei, Z., Yang, Y., Wu, Z.: Ensemble of Support Vector Machine for Text-Independent Speaker Recognition. International Journal of Computer Science and Network Security (May 2006)
10. Lin, H.-T., Li, L.: Infinite ensemble learning with support vector machines. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 242–254. Springer, Heidelberg (2005)
11. Dong, Y.-S., Han, K.-S.: Text Classification Based On Data Partitioning and Parameter Varying Ensemble. In: SAC 2005 (March 13-17, 2005)
12. Perdisci, R., Gu, G., Lee, W.: Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection System. In: Perner, P. (ed.) ICDM 2006. LNCS (LNAI), vol. 4065. Springer, Heidelberg (2006)
13. Zhou, Z.H., Li, M.: Tri-Training: Exploiting unlabeled data using three classifiers. IEEE Trans. on Knowledge and Data Engineering 17(11), 1529–1541 (2005)
14. ChapelleVikas, O., Keerthi, S.S.: Optimization Techniques for Semi-Supervised Support Vector Machines. Journal of Machine Learning Research 9, 203–233 (2008)
15. Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines. In: International Conference on Machine Learning, ICML (1999)
16. Stepenosky, N., Green, D., Kounios, J., Clark, C.M., Polikar, R.: Majority Vote and Decision Template Based Ensemble Classifiers Trained on Event Related Potentials for Early Diagnosis of Alzheimer's Disease. In: ICASSP 2006 (2006)

Kernels Based on Distributions of Agreement Subtrees

Kilho Shin[1] and Tetsuji Kuboyama[2]

[1] Carnegie Mellon CyLab Japan
[2] Gakushuin University

Abstract. The MAST (maximum agreement subtrees) problem has been extensively studied, and the size of the maximum agreement subtrees between two trees represents their similarity. This similarity measure, however, only takes advantage of a very small portion of the agreement subtrees, that is, the maximum agreement subtrees, and agreement subtrees of smaller size are neglected at all. On the other hand, it is reasonable to consider that the distributions of the sizes of the agreement subtrees may carry useful information with respect to similarity. Based on the notion of the *size-of-index-structure-distribution kernel* introduced by Shin and Kuboyama, the present paper introduces positive semidefinite tree-kernels, which evaluate distributional features of the sizes of agreement subtrees, and shows efficient dynamic programming algorithms to calculate the kernels. In fact, the algorithms are of $O(|x| \cdot |y|)$-time for labeled and ordered trees x and y. In addition, the algorithms are designed so that the agreement subtrees have roots and leaves with labels from predetermined sub-domains of an alphabet. This design will be very useful for important applications such as the XML documents.

1 Introduction

When some structures are commonly derived from two data objects, the structures may carry information with regard to similarities between the data objects. In [1], such structures are referred to as *index structures*.

The *agreement subtree* is a good example of the index structure, when data objects are represented as trees. An agreement subtree between plural input trees is usually defined as a subtree *homeomorphically* included in all the input trees ([3] and Definition 4 and 5), and the maximum size of the possible agreement subtrees can be naturally viewed as a measure of similarity between the input trees. Thus, the *maximum agreement subtrees* (MAST) problem is defined as the problem to determine at least one agreement subtree with the *maximum size* among the possible agreement subtrees for the input trees. The maximum agreement subtree has been used in a wide range of application (*e.g.* evolutionary trees [4,3]).

From the algorithm efficiency point of view, the MAST problem is NP-hard on three rooted trees of unbounded degree [5]. The degree of a tree is defined as

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 236–246, 2008.

the maximum of the number of the children of a single vertex (node) of the tree. On the other hand, when the degree is bounded by d, an $O(\sqrt{d}n \log \frac{2n}{d})$-time algorithm is known on two rooted n-leaf trees [6]. For unrooted trees, we have an $O(n^{1.5})$-time algorithm [7].

On the other hand, Shin and Kuboyama [1] introduced the notion of the *size-of-index-structure-distribution* kernel, which is a class of positive semidefinite kernels that evaluate the *distributions* of the sizes of index structures.

In particular, when the agreement subtree is used as the index structure, the corresponding size-of-index-structure-distribution kernel is defined as follows for *labeled*, *ordered* and *rooted* trees x and y.

$$K(x, y) = \sum_{t \in \mathsf{AST}(x,y)} f(\mathsf{size\_of}(t)) \tag{1}$$

In Eq.(1), $\mathsf{AST}(x, y)$ denotes the set of all the agreement subtrees between x and y, and f is an arbitrary function $f : \mathbb{N} \to \mathbb{R}_+ = \{y \geq 0 \mid y \in \mathbb{R}\}$. Shin and Kuboyama [1] also claimed that there exist polynomial-time algorithms to evaluate $K(x, y)$ when x and y are of *bounded* degree and $f(x)$ is either $f(x) = \alpha^x$ or $f(x) = x$, although they didn't show the algorithms.

As claimed in [1], the advantage of $K(x, y)$ of Eq.(1) over $\max\{\mathsf{size\_of}(t) \mid t \in \mathsf{AST}(x, y)\}$ as a similarity measure between x and y is that $K(x, y)$ takes into account those agreement subtrees smaller in size than the maximum ones.

Based on the discussion so far, the contribution of the present paper can be summarized as follows.

First, we present $O(|x| \cdot |y|)$-time dynamic programming algorithms to calculate $K(x, y)$ of Eq.(1) for the functions of $f(x) = x$ and $f(x) = \alpha^x$. What is to be remarked here is that we don't assume that x and y are of bounded degree. Shin and Kuboyama [1] claimed the existence of polynomial-time algorithms only for x and y of bounded degree, and also, no polynomial time algorithm for MAST is known when the input rooted trees x and y are of unbounded degree.

Secondly, we add certain flexibility to the notions of agreement subtrees and their sizes. In fact, for three alphabets Σ_{start}, Σ_{end} and Σ_{count}, which may be different from each other, $\mathsf{AST}(x, y)$ is defined as the set of the agreement subtrees such that their roots and leaves are respectively labeled by elements of Σ_{start} and Σ_{end}, and the size of an agreement subtree t is defined as the number of the vertices of t whose labels belong to Σ_{count}.

Also, through experiments, we see that our kernels with $f(x) = \alpha^x$ for appropriate α show better performance than the elastic tree kernel by Kashima and Koyanagi [2], which corresponds to the case of $f(x) \equiv 1$.

2 Agreement Subtrees

Kao et al. [6] introduced a generalized setting for agreement subtrees and their sizes for *unrestricted* labeled trees: all the vertices of input trees and agreement subtrees are labeled by elements of an alphabet Σ, and, when an agreement subtree is *homeomorphically* embedded into an input tree, we impose the constraint

that the labels of the corresponding vertices of the agreement tree and the input tree coincide. Furthermore, a subset $\Sigma_{\mathsf{count}} \subseteq \Sigma$ is given, the size of an agreement subtree is defined as the number of the vertices whose labels belong to Σ_{count}. This setting covers the conventional setting for evolutionary trees (*e.g.* [4,3]), where only leaves are labeled and the size of an agreement subtree is defined as the number of its leaves: we have only to let $\Sigma = \Sigma_{\mathsf{count}} \cup \{\lambda\}$ and to label all the non-leaf vertices with λ.

In the present paper, we modify the setting by Kao et al. by introducing two more subsets of the alphabet Σ, namely Σ_{start} and Σ_{end}, and impose the following constraints on agreement subtrees, which will be very useful for applications in the real world.

- The root of an agreement subtree shall be labeled with an element of Σ_{start}.
- Each leaf of an agreement subtree shall be labeled with an element of Σ_{end}.

In the remainder of this section, we will give the definitions and notations to be used in the present paper.

Definition 1 (Tree). *A* tree *means a rooted, ordered and labeled tree.*

1. *The tree has a* root*, which is an ancestor of all the other vertices.*
2. *In addition to the hierarchical (ancestor-descendent) partial order, a left-to-right (sibling) partial order are given to the set of the vertices of the tree.*
3. *Each vertex v of the tree is labeled with an element of an alphabet Σ.*

When a vertex v is an ancestor of (left to) a vertex y, we denote the relation by $v > w$ ($v \prec w$, resp.). Also, $\ell_v \in \Sigma$ denotes the label associated with v.

Definition 2 (Forest). *A forest is a tree without the root, and eventually a sequence of trees.*

Definition 3 (Nearest common ancestor and $v \smile w$). *Let x be a forest, and let v and w be vertices of x. The* nearest common ancestor *$v \smile w$ of v and w, if present, is the vertex that satisfies the following conditions.*

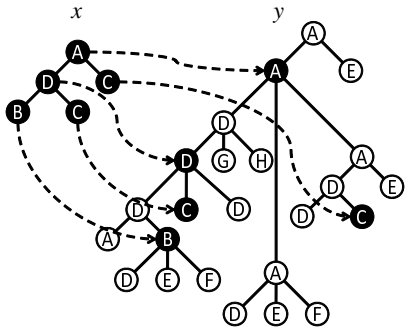
- *$v \leq v \smile w$ and $w \leq v \smile w$.*
- *If $v \leq u$ and $w \leq u$, then $v \smile w \leq u$.*

Definition 4 (Homeomorphic embedding). *Let x and y be respectively a tree or a forest, and let φ be a one-to-one mapping from the set of the vertices of x to that of y. The mapping φ* homeomorphically *embeds x into y, if, and only if, the following conditions are met for arbitrary vertices v and w of x.*

1. *If $v \smile w$ exists, $\varphi(v \smile w) = \varphi(v) \smile \varphi(w)$ holds.*
2. *If $v \prec w$ holds, $\varphi(v) \prec \varphi(w)$ holds.*
3. *$\ell_v = \ell_{\varphi(v)}$ holds.*

Also, φ is called a homeomorphic embedding*.*

Proposition 1 immediately follows from Definition 4.



The figure depicts a homeomorphic embedding from the tree x into the tree y. The character in each circle represent the label of the vertex. Furthermore, this homeomorphic embedding is the unique homeomorphic embedding from x to y. For example, by definition of the homeomorphic embedding, the root of x must map to a vertex in y such that:

- it has the label A and
- it is the nearest common ancestor of two vertices with label C and D.

Such a vertex is uniquely determined as seen in the figure.

Fig. 1. A homeomorphic embedding

Proposition 1. *Let x be a tree or a forest, and x' be a subset of vertices of x. Then, the following conditions are equivalent to each other.*

1. *There exists a tree t and a homeomorphic embedding $\varphi : t \to x$ such that $\varphi(t) = x'$.*
2. *x' is* totally *closed with respect to the nearest-common-ancestor operator $\smile$: for arbitrary $v \in x'$ and $w \in x'$, $v \smile w$ exists, and falls into x'*

Definition 5 (Agreement subtree and $\mathsf{AST}(x, y)$). *Let x and y be respectively a tree or a forest. A tree t is called an* agreement subtree *between x and y, if, and only if, the following conditions are met.*

- *t is homeomorphically embedded into both x and y.*
- *The root of t is labeled with an element of Σ_{start}.*
- *The leaves of t are labeled with elements of Σ_{end}.*

In addition, $\mathsf{size\_of}(t)$ is defined as the number of the vertices of t whose labels belong to Σ_{count}, and the set of all the agreement subtrees between x and y is denoted by $\mathsf{AST}(x, y)$.

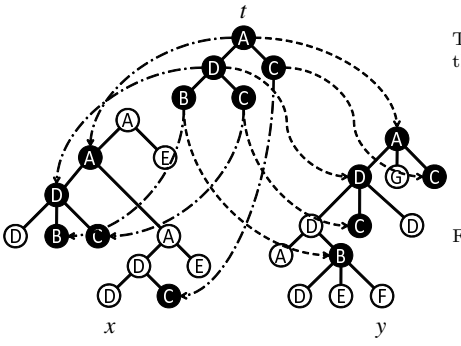
3 Size-of-Agreement-Subtree-Distribution Kernels

For an arbitrary non-negative function $f : \mathbb{N} \to \mathbb{R}_+$, Shin and Kuboyama [1] proved that the bivariable function $\mathsf{K}^{[f]}$ defined by Eq.(2) is positive semidefinite.

$$\mathsf{K}^{[f]}(x, y) = \sum_{t \in \mathsf{AST}(x,y)} f(\mathsf{size\_of}(t)) \tag{2}$$

Not necessarily, the positive semidefinite kernel $\mathsf{K}^{[f]}$ can be evaluated efficiently. In the present paper, we focus on the cases of $f(x) = \alpha^x$ and $f(x) = x$, and show efficient dynamic programming algorithms for the cases.

$$\mathsf{K}^{[\alpha^x]}(x, y) = \sum_{t \in \mathsf{AST}(x,y)} \alpha^{\mathsf{size\_of}(t)}, \quad \mathsf{K}^{[x]}(x, y) = \sum_{t \in \mathsf{AST}(x,y)} \mathsf{size\_of}(t)$$



The figure depicts an agreement tree pair under the following setting.

$$\Sigma = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F}, \mathsf{G}, \mathsf{H}\}$$
$$\Sigma_{\mathsf{start}} = \{\mathsf{A}\}$$
$$\Sigma_{\mathsf{end}} = \{\mathsf{B}, \mathsf{C}, \mathsf{D}\}$$
$$\Sigma_{\mathsf{count}} = \{\mathsf{D}, \mathsf{E}, \mathsf{F}\}$$

Furthermore, we have the following.

$$\mathsf{size\_of}(t) = 1$$

Fig. 2. An agreement subtree

4 The Recursive Formulas

In this section, we present recursive formulas to evaluate $\mathsf{K}^{[\alpha^x]}(x, y)$ and $\mathsf{K}^{[x]}(x, y)$. Since dynamic programming algorithms are derived from the formulas in a straightforward manner, we will not show the algorithms in the present paper.

Before proceeding, we will introduce a couple of symbols to be used in the recursive formulas (Fig. 3).

- When f_1 and f_2 are respectively a tree or a forest, $f_1 \bullet f_2$ is the forest obtained by appending f_2 to f_1 at the rightmost position. In particular, if f consists of n trees $t_1, \ldots, t_n$ in the left-to-right direction, we have the expression $f = t_1 \bullet t_2 \bullet \cdots \bullet t_n$.
- For a vertex v and a forest (tree) f, $v \circ f$ denotes the tree such that its root is v and the forest determined by eliminating the root is f.

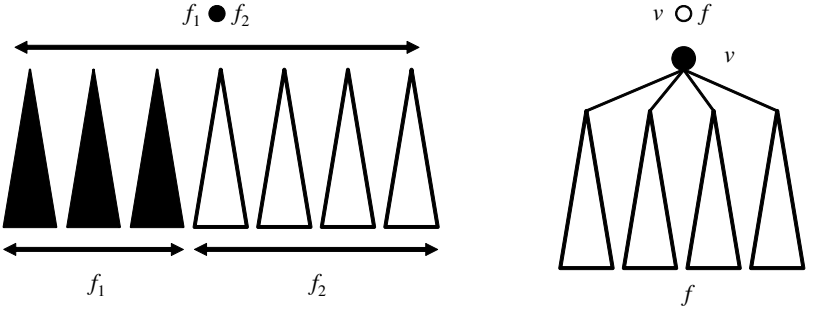


Fig. 3. Concatenation and aggregation

4.1 The Case of $f(x) = \alpha^x$

In order to evaluate $\mathsf{K}^{[\alpha^x]}(x, y)$, we will introduce two more bivariable functions, namely $\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(x, y)$ and $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(x, y)$.

- The function $\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(x, y)$ is the same as the $\mathsf{K}^{[\alpha^x]}(x, y)$ except that $\Sigma_{\mathsf{start}} = \Sigma$ is assumed for $\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(x, y)$.
- We let $\mathsf{ASF}(x, y)$ be the set of the forests f such that:
  - f is homeomorphically embedded into x and y by φ and ψ, and
  - if $\varphi(v) \smile \varphi(w)$ or $\psi(v) \smile \psi(w)$ exists for $v, w \in f$, $v \smile w$ exists as well.

  Then, $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(x, y)$ is defined by Eq.(3).

$$\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(x, y) = \sum_{f \in \mathsf{ASF}(x,y)} \alpha^{\mathsf{size\_of}(f)} \tag{3}$$

Remark 1. Let $\varphi : f \to x$ be a homeomorphic embedding onto x''. If $v \smile w$ exists for $v, w \in f$, $\varphi(v) \smile \varphi(w)$ necessarily exists, and falls into x''. In contrast, the existence of $\varphi(v) \smile \varphi(w)$ doesn't necessarily indicates that of $v \smile w$. If $v \smile w$ doesn't exist, $\varphi(v) \smile \varphi(w)$ doesn't falls into x''. Thus, x'' is not necessarily closed with respect to $\smile$ (compare with Proposition 1).

Remark 2. The following formula holds for $f = t_1 \bullet \cdots \bullet t_m$ and $f' = t'_1 \bullet \cdots \bullet t'_{m'}$.

$$\begin{aligned}
&\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f, f') = \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(t_1 \bullet \cdots \bullet t_m, t'_1 \bullet \cdots t'_{m'}) \\
&= \sum_{n=1}^{\min\{m,m'\}} \left(\sum_{1 \le i_1 < \cdots < i_n \le m} \left(\sum_{1 \le i'_1 < \cdots < i'_n \le m'} \prod_{j=1}^{n} \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(t_{i_j}, t'_{i'_j}) \right) \right)
\end{aligned}$$

The recursive formulas should include termination conditions and reduction formulas, and Eq.(4) gives the termination conditions.

$$\begin{aligned}
&\mathsf{K}^{[\alpha^x]}(x, \emptyset) = \mathsf{K}^{[\alpha^x]}(\emptyset, x) = \\
&\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(x, \emptyset) = \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(\emptyset, x) = \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(x, \emptyset) = \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(\emptyset, x) = 0
\end{aligned} \tag{4}$$

The reduction formulas are introduced as follows.

When at least one of x and y is a *proper* forest, we let $x = t_1 \bullet f_1$ and $y = t_2 \bullet f_2$, where t_1 and t_2 are the leftmost trees of x and y, respectively. For $t \in \mathsf{AST}(x, y)$, which is embedded into x and y by φ and ψ, the root of $x' = \varphi(t)$ ($y' = \psi(t)$) is located in either t_1 or f_1 (either t_2 or f_2, resp.). Therefore, the reduction formula for $\mathsf{K}^{[\alpha^x]}(x, y)$ is given by Eq.(5).

$$\begin{aligned}
\mathsf{K}^{[\alpha^x]}(t_1 \bullet f_1, t_2 \bullet f_2) = {} & \mathsf{K}^{[\alpha^x]}(t_1, t_2) + \mathsf{K}^{[\alpha^x]}(t_1, f_2) + \\
& \mathsf{K}^{[\alpha^x]}(f_1, t_2) + \mathsf{K}^{[\alpha^x]}(f_1, f_2)
\end{aligned} \tag{5}$$

When both of x and y are trees, we let $x = v_1 \circ f_1$ and $y = v_2 \circ f_2$, and we have the following four cases.

(a) The roots of x' and y' are v_1 and v_2, respectively.
(b) The root of x' is v_1, and that of y' is not v_2.
(c) The root of x' is not v_1, and that of y' is v_2.
(d) The root of x' is not v_1, and that of y' is not v_2.

Case (a) occurs, only when the labels of v_1 and v_2 are identical to each other and belong to Σ_{start} ($\ell_{v_1} = \ell_{v_2} \in \Sigma_{\mathsf{start}}$). Further, when we let $x' = v_1 \circ x''$ and $y' = v_1 \circ y''$, the case (a) is to be subdivided into subcases according to what are x'' and y''. Note that x'' and y'' are *homeomorphic* to each other.

(a.1) $x'' = y'' = \emptyset$
(a.2) $x'' \neq \emptyset$ and $y'' \neq \emptyset$

In Case (a.1), the contribution of (x', y') to $\mathsf{K}^{[\alpha^x]}(v_1 \circ f_1, v_2 \circ f_2)$ is evaluated as follows.

$$\text{The contribution} = \begin{cases} 1 & \text{if } \ell_{v_1} = \ell_{v_2} \in \Sigma_{\mathsf{start}} \cap \Sigma_{\mathsf{end}} \setminus \Sigma_{\mathsf{count}}, \\ \alpha & \text{if } \ell_{v_1} = \ell_{v_2} \in \Sigma_{\mathsf{start}} \cap \Sigma_{\mathsf{end}} \cap \Sigma_{\mathsf{count}}, \\ 0 & \text{otherwise.} \end{cases}$$

Then, when we define $\mathsf{factor}(\Sigma_{\mathsf{count}}, \alpha : v_1, v_2)$ as follows, the contribution is evaluated by $\mathsf{factor}(\Sigma_{\mathsf{count}}, \alpha : v_1, v_2) \cdot \mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{start}}] \cdot \mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{end}}]$.

$$\mathsf{factor}(\Sigma_{\mathsf{count}}, \alpha : v_1, v_2) = \begin{cases} 0, & \text{if } \ell_{v_1} \neq \ell_{v_2}, \\ 1, & \text{if } \ell_{v_1} = \ell_{v_2} \notin \Sigma_{\mathsf{count}}, \\ \alpha, & \text{if } \ell_{v_1} = \ell_{v_2} \in \Sigma_{\mathsf{count}}. \end{cases}$$

$$\mathsf{eval}[\mathit{predicate}] = \begin{cases} 1, & \text{if } \mathit{predicate} \text{ is true,} \\ 0, & \text{if } \mathit{predicate} \text{ is false.} \end{cases}$$

In Case (a.2), x'' and y'' have the following property.

Lemma 1. *x'' and y'' are closed with respect to the nearest-common-ancestor operator $\smile$ in f_1 and f_2.*

Proof. Let v and w be vertices of x''. By Proposition 1, The vertex $v \smile w \in v_1 \circ f_1$ is in x'. Therefore, $v \smile w$ is identical to v_1 or falls in x''. ■

Therefore, the contribution of Case (a.2) to $\mathsf{K}^{[\alpha^x]}(v_1 \circ f_1, v_2 \circ f_2)$ is evaluated by $\mathsf{factor}(\Sigma_{\mathsf{count}}, \alpha : v_1, v_2) \cdot \mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{start}}] \cdot \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)$.

The contributions of Case (b), (c) and (d) are respectively evaluated by:

(b) $\mathsf{K}^{[\alpha^x]}(v_1 \circ f_1, f_2) - \mathsf{K}^{[\alpha^x]}(f_1, f_2)$,
(c) $\mathsf{K}^{[\alpha^x]}(f_1, v_2 \circ f_2) - \mathsf{K}^{[\alpha^x]}(f_1, f_2)$ and
(d) $\mathsf{K}^{[\alpha^x]}(f_1, f_2)$.

Finally, we have Eq.(6) for the case where x and y are trees.

$$\begin{aligned}\mathsf{K}^{[\alpha^x]}(v_1 \circ f_1, v_2 \circ f_2) = \mathsf{factor}(\Sigma_{\mathsf{count}}, \alpha : v_1, v_2) \cdot \mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{start}}] \cdot \\ \Big(\mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{end}}] + \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)\Big) + \\ \mathsf{K}^{[\alpha^x]}(v_1 \circ f_1, f_2) + \mathsf{K}^{[\alpha^x]}(f_1, v_2 \circ f_2) - \mathsf{K}^{[\alpha^x]}(f_1, f_2) \end{aligned} \tag{6}$$

For $\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(x, y)$, we can derive Eq.(7) and (8) in the same way as for $\mathsf{K}^{[\alpha^x]}(x, y)$.

$$\begin{aligned}&\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(t_1 \bullet f_1, t_2 \bullet f_2) \\ &= \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(t_1, t_2) + \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(t_1, f_2) + \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(f_1, t_2) + \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(f_1, f_2)\end{aligned} \tag{7}$$

$$\begin{aligned}\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(v_1 \circ f_1, v_2 \circ f_2) = \mathsf{factor}(\Sigma_{\mathsf{count}}, \alpha : v_1, v_2)\Big(\mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{end}}] + \\ \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)\Big) + \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(v_1 \circ f_1, f_2) + \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(f_1, v_2 \circ f_2) - \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(f_1, f_2)\end{aligned} \tag{8}$$

To evaluate $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(x, y)$ for $x = t_1 \bullet f_1$ and $y = t_2 \bullet f_2$, we consider the following cases. For $f \in \mathsf{ASF}(x, y)$, which is embedded into x and y by φ and ψ, we let $x'' = \varphi(f)$ and $y'' = \psi(f)$.

(a) $x'' \cap t_1 \neq \emptyset$ and $y'' \cap t_2 \neq \emptyset$
(b) $x'' \cap t_1 = \emptyset$ and $y'' \cap t_2 \neq \emptyset$
(c) $x'' \cap t_1 \neq \emptyset$ and $y'' \cap t_2 = \emptyset$
(d) $x'' \cap t_1 = \emptyset$ and $y'' \cap t_2 = \emptyset$

The contributions of Case (a), (b), (c) and (d) to $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(t_1 \bullet f_1, t_2 \bullet f_2)$ are respectively evaluated as follows.

(a) $\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(t_1, t_2)\Big(1 + \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)\Big)$, since $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(t_1, t_2) = \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(t_1, t_2)$.
(b) $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, t_2 \bullet f_2) - \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)$.
(c) $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(t_1 \bullet f_1, f_2) - \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)$.
(d) $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)$.

Hence, we have Eq.(9) as the reduction formula for $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(x, y)$.

$$\begin{aligned}\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(t_1 \bullet f_1, t_2 \bullet f_2) = \mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}(t_1, t_2)\Big(1 + \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)\Big) \\ + \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, t_2 \bullet f_2) + \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(t_1 \bullet f_1, f_2) - \mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}(f_1, f_2)\end{aligned} \tag{9}$$

In a straightforward manner, a dynamic programming algorithm is derived from Eq.(4) to Eq.(9).

Proposition 2. *The dynamic program derived from Eq.(4) to Eq.(9) to evaluate* $\mathsf{K}^{[\alpha^x]}(x, y)$ *terminates.*

Proof. If at least one of x and y is a forest, the reduction formulas Eq.(5), Eq. (7) and Eq. (9) reduce the width (*i.e.* the number of tree components) of the forest.

On the other hand, if x and y are both trees, the reduction formulas Eq.(6) and Eq.(8) reduce the height of the trees. ∎

In particular, when $\alpha = 1$, $\mathsf{K}^{[\alpha^x]}(x, y) = \mathsf{K}^{[1^x]}(x, y)$ represents the number of the agreement subtree between x and y (*i.e.* $|\mathsf{AST}(x, y)|$).

4.2 The Case of $f(x) = x$

The termination conditions and the reduction formulas for $f(x) = x$ are derived in the same way as for $f(x) = \alpha^x$.

$$\mathsf{K}^{[x]}(x, \emptyset) = \mathsf{K}^{[x]}(\emptyset, x) = \mathsf{K}^{[x]}_{\mathsf{tree}}(x, \emptyset) = \mathsf{K}^{[x]}_{\mathsf{tree}}(\emptyset, x) = \mathsf{K}^{[x]}_{\mathsf{forest}}(x, \emptyset) = \mathsf{K}^{[x]}_{\mathsf{forest}}(\emptyset, x) = 0$$

$$\mathsf{K}^{[x]}(t_1 \bullet f_1, t_2 \bullet f_2) = \mathsf{K}^{[x]}(t_1, t_2) + \mathsf{K}^{[x]}(t_1, f_2) + \mathsf{K}^{[x]}(f_1, t_2) + \mathsf{K}^{[x]}(f_1, f_2)$$

$$\begin{aligned}\mathsf{K}^{[x]}(v_1 \circ f_1, v_2 \circ f_2) &= \mathsf{eval}[\ell_{v_1} = \ell_{v_2}] \cdot \mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{start}}] \cdot \\ &\Big\{\mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{count}}] \Big(\mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{end}}] + \mathsf{K}^{[1^x]}_{\mathsf{forest}}(f_1, f_2)\Big) + \mathsf{K}^{[x]}_{\mathsf{forest}}(f_1, f_2)\Big\} + \\ &\mathsf{K}^{[x]}(v_1 \circ f_1, f_2) + \mathsf{K}^{[x]}(f_1, v_2 \circ f_2) - \mathsf{K}^{[x]}(f_1, f_2)\end{aligned}$$

$$\mathsf{K}^{[x]}_{\mathsf{tree}}(t_1 \bullet f_1, t_2 \bullet f_2) = \mathsf{K}^{[x]}_{\mathsf{tree}}(t_1, t_2) + \mathsf{K}^{[x]}_{\mathsf{tree}}(t_1, f_2) + \mathsf{K}^{[x]}_{\mathsf{tree}}(f_1, t_2) + \mathsf{K}^{[x]}_{\mathsf{tree}}(f_1, f_2)$$

$$\begin{aligned}\mathsf{K}^{[x]}_{\mathsf{tree}}(v_1 \circ f_1, v_2 \circ f_2) &= \mathsf{eval}[\ell_{v_1} = \ell_{v_2}] \cdot \\ &\Big\{\mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{count}}] \Big(\mathsf{eval}[\ell_{v_1} \in \Sigma_{\mathsf{end}}] + \mathsf{K}^{[1^x]}_{\mathsf{forest}}(f_1, f_2)\Big) + \mathsf{K}^{[x]}_{\mathsf{forest}}(f_1, f_2)\Big\} + \\ &\mathsf{K}^{[x]}_{\mathsf{tree}}(v_1 \circ f_1, f_2) + \mathsf{K}^{[x]}_{\mathsf{tree}}(f_1, v_2 \circ f_2) - \mathsf{K}^{[x]}_{\mathsf{tree}}(f_1, f_2)\end{aligned}$$

$$\begin{aligned}\mathsf{K}^{[x]}_{\mathsf{forest}}(t_1 \bullet f_1, t_2 \bullet f_2) &= \mathsf{K}^{[x]}_{\mathsf{tree}}(t_1, t_2)\Big(1 + \mathsf{K}^{[1^x]}_{\mathsf{forest}}(f_1, f_2)\Big) + \\ &\mathsf{K}^{[1^x]}_{\mathsf{tree}}(t_1, t_2)\mathsf{K}^{[x]}_{\mathsf{forest}}(f_1, f_2) + \mathsf{K}^{[x]}_{\mathsf{forest}}(f_1, t_2 \bullet f_2) + \mathsf{K}^{[x]}_{\mathsf{forest}}(t_1 \bullet f_1, f_2) - \mathsf{K}^{[x]}_{\mathsf{forest}}(f_1, f_2)\end{aligned}$$

5 Evaluation of the Efficiency

Proposition 3. *The dynamic programs for $\mathsf{K}^{[\alpha^x]}(x, y)$ and $\mathsf{K}^{[x]}(x, y)$ derived from the aforementioned formulas run in $O(|x| \cdot |y|)$-time.*

We will enumerate the subforest pairs (x', y') that are to be input into $\mathsf{K}^{[\alpha^x]}$, $\mathsf{K}^{[\alpha^x]}_{\mathsf{tree}}$, $\mathsf{K}^{[\alpha^x]}_{\mathsf{forest}}$, $\mathsf{K}^{[x]}$, $\mathsf{K}^{[x]}_{\mathsf{tree}}$ and $\mathsf{K}^{[x]}_{\mathsf{forest}}$. When we denote the subtree of x consisting of the vertices that are below or equal to v by $x[v]$, x' is one of the following.

- For a vertex v in x, x' is $x[v]$.
- For a vertex v in x, x' is $x[v] \bullet x[v_1] \bullet \cdots \bullet x[v_n]$, where $(v_1, \ldots, v_n)$ is the sequence of the sibling vertices (sharing the same parent vertex) of v such that $v \prec v_i$.

Apparently, the number of such x' is $O(|x|)$.

6 Experimental Results

In this section, we test the size-of-agreement-subtree-distribution kernel with SVM by empirically comparing its predictive performance in glycan structure classification problem. Glycans are carbohydrate chains that are considered to play an important role in various fundamental biological processes such as cell-cell interaction. The structure of a glycan is abstractly represented as a tree structure by representing single carbohydrates as nodes and their covalent bond as edges.

The glycan structures are retrieved from the KEGG/GLYCAN database [10], and the annotations are from the CarbBank/CCSD database [11]. In these annotations, we employ two blood components, *leukemic cells*, and the other non-leukemic blood components (*erythrocyte*, *serum*, and *plasma*). The glycan structures of leukemic cells are served as positive training examples. The numbers of positive and negative data are respectively 192 and 294. The tree structures obtained from the glycan data include 29 kinds of node labels, while all the edge labels are omitted for simplicity.

Our kernel was implemented in Ruby and executed on a Linux machine, and was normalized by $K(x,y)/(\sqrt{K(x,x)}\sqrt{K(y,y)})$. We used LIBSVM ([9]) as the SVM implementation, and used the area under the ROC curve (AUC) as the performance measure. The ROC curve plots the relationship between the true positive rate and the false positive rate. The AUC is the prevailing performance measure for a decision function with a kernel that separates positive examples from negative ones. The AUC values range from 0.5 to 1.0, where the value 0.5 indicates a random separation and the value 1.0 indicates a perfect separation. All of the performance measures are calculated with 5-fold cross validation.

Figure 4 shows the predictive performance of our kernel with function $f(x) = \alpha^x$ while varying the parameter α, and with function $f(x) = x$ (indicated by 'χ' in Figure 4). Since $\mathsf{K}^{[\alpha^x]}(x,y)$ for $\alpha = 1$ coincides with the elastic tree kernel [2], from Figure 4) presents that $\mathsf{K}^{[\alpha^x]}(x,y)$ for $\alpha < 1$ showed better performance

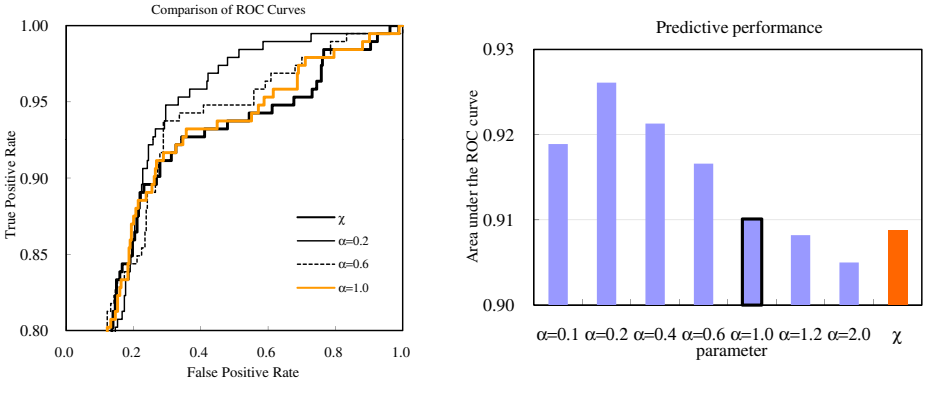


Fig. 4. ROC curves and areas under the ROC curves for glycan classification

than the elastic tree kernel. On the other hand, $\mathsf{K}^{[x]}(x, y)$ is only compatible with the elastic tree kernel.

7 Conclusion

We have presented a novel approach to designing kernels for trees, which evaluate distributional features of the sizes of agreement subtrees between two trees. Moreover, we have developed quadratic-time algorithms to calculate the kernels. By applying our kernel to a glycan classification problem, we have shown the effectiveness of our approach.

Acknowledgements

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), 20500126, 2008.

References

1. Shin, K., Kuboyama, T.: A generalization of haussler's convolution kernel - mapping kernel. In: Proc. of The 25th International Conference On Machine Learning, ICML (2008)
2. Kashima, H., Koyanagi, T.: Kernels for Semi-Structured Data. In: Proc. of The 9th International Conference on Machine Learning (ICML), pp. 291–298 (2002)
3. Berry, V., Nicolas, F.: Maximum Agreement and Compatible Supertrees (Extended Abstract). In: Sahinalp, S.C., Muthukrishnan, S.M., Dogrusoz, U. (eds.) CPM 2004. LNCS, vol. 3109, pp. 205–219. Springer, Heidelberg (2004)
4. Hein, J., Jiang, T., Wang, L., Zhang, K.: On the complexity of comparing evolutionary trees. Discrete Applied Mathematics 71, 153–169 (1996)
5. Amir, A., Keselman, D.: Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithm. SIAM J. Computing 26(6), 1656–1669 (1997)
6. Kao, M.Y., Lam, T.W., Sung, W.-K., Ting, H.F.: An even faster and more unifying algorithm for comparing trees via unbalanced bipartite matching. J. Algorithms 40(2), 212–233 (2001)
7. Kao, M.Y., Lam, T.W., Sung, W.-K., Ting, H.F.: A decomposition theorem for maximum weight bipartite matching with applications to evolutionary trees. SIAM J. Computing 31(1), 18–26 (2001)
8. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, Cambridge (2000)
9. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines, Software (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
10. Hashimoto, K., Goto, S., Kawano, S., Aoki-Kinoshita, K.F., Ueda, N.: KEGG as a glycome informatics resource. Glycobiology 16, 63R–70R (2006)
11. Doubet, S., Albersheim, P.: CarbBank. Glycobiology 2(6), 505 (1992)
12. Yamanishi, Y., Bach, F., Vert, J.-P.: Glycan classification with tree kernels. Bioinformatics 23(10), 1211–1216 (2007)

Practical Bias Variance Decomposition

Remco R. Bouckaert

Computer Science Department, University of Waikato, New Zealand
rrb@xm.co.nz, remco@cs.waikato.ac.nz

Abstract. Bias variance decomposition for classifiers is a useful tool in understanding classifier behavior. Unfortunately, the literature does not provide consistent guidelines on how to apply a bias variance decomposition. This paper examines the various parameters and variants of empirical bias variance decompositions through an extensive simulation study. Based on this study, we recommend to use ten fold cross validation as sampling method and take 100 samples within each fold with a test set size of at least 2000. Only if the learning algorithm is stable, fewer samples, a smaller test set size or lower number of folds may be justified.

1 Introduction

Is the improved performance of C4.5 with increasing training set sizes due to decrease in bias, decrease in variance, or both? Figure 1 shows how bias variance decomposition changes with increasing training set size. The first three plates are three separate runs of decompositions according to Kohavi [1] (using Weka [2] with default settings, i.e. 50 samples). Everything was kept the same except that a different randomization of the data set was used. The first plate suggests both variance and bias decreases with increasing training set size. The second plate suggests bias decreases but variance remains the same. The third plate suggests bias remains unchanged while variance decreases. This raises the following questions: why are these decompositions so different and how to select the correct decomposition.

The reason for the conclusions of these decompositions to be so radically different is because the decomposition tends to be sensitive to the particular randomization of the data set. Clearly, this is not desirable. Furthermore, the example shown in Figure 1 is not a hand picked example or fluke of the data. The plates shown are selected from among the first ten runs and the dataset was generated from a naive Bayes data source (see Section 4). It did not take long to find such contradicting outcomes.

Clearly, it is important for a bias variance decomposition to show low variability of the estimates, since it impacts conclusions drawn from them. The randomization of the dataset is not the only issue that impacts the variability of the decomposition. In the literature (Section 2.1), various methods for bias variance decomposition are proposed, but no two papers seem to agree on the parameters of an empirical method. In this paper, we investigate these issues and

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 247–257, 2008.

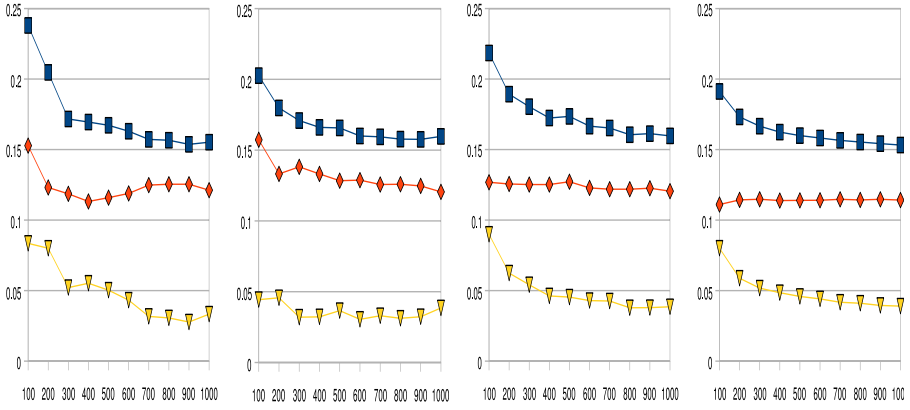


Fig. 1. Various bias variance decompositions on the same problem (see text for description). Training set size on x-axis, error (top line) bias (middle line) and variance (bottom line) on y-axis.

perform an empirical investigation in order to be able to recommend a sound way to perform bias variance decompositions that minimizes variability of the bias and variance estimates. Based on our findings, the fourth plate in Figure 1 turns out to be the correct interpretation.

2 Bias Variance Decomposition

Bias variance decomposition described in the literature follows the following pattern (illustrated in Figure 2). A *data source* is a provider of *training sets*. Data sources can be synthetic where a model of the domain is used to create new independent training sets every time a new set is requested. In practice, we are more interested in data sources that take samples of fixed data sets. The training sets obtained from the data source are used to train a learning algorithm. The thus leaned models are applied on a *test set*. The bias and variance are then estimated for each instance in the test set and for 0-1 loss the bias and variance are calculated from the number of incorrect learners for the instance. The bias for an instance is estimated as $\sum_i (x_i - p_i)^2 - p_i * (1 - p_i)/(n - 1)$ where i sums over the class values, n is the number of learners applied to the instance, x_i an indicator variable (0 or 1) that is indicates whether the instance class value equals the ith value, and p_i the fraction of learners that correctly predicted x_i. The variance for an instance is estimated as $1 - \sum_i p_i^2$. The final bias and variance reported are averaged over the instances in the test set.

2.1 Experimental Methodology from the Literature

In the literature, the following methods for bias variance decomposition can be found.

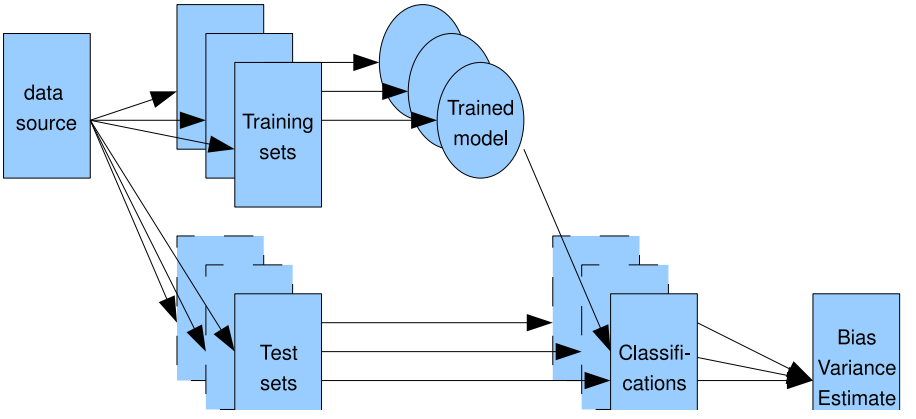


Fig. 2. General overview of bias variance decomposition estimation

Kong and Dietterich [3] use a synthetic data source for 200 data sets of 200 instances and a test set of 7670 instances were drawn for direct estimates of bias and variance.

In Kohavi and Wolpert [1], a sample of size n without replacement from a data set D is taken to get a training set source. From the remainder, a test set of size n is sampled without replacement. The training source set is used to draw a sample of size $m < n$ on which the learning algorithm is trained. There are 50 such training samples drawn. The resulting 50 trained models are then applied to the test set and the bias and variance of the learner are calculated from the predictions on the test set.

Domingos [4] splits data sets randomly into training and test sets with a proportion of 2:1. From the training set 100 bootstrap samples (i.e. samples with replacement) were taken and the learner trained on the 100 samples and applied to the test set.

Bauer and Kohavi [5] use the technique of [4] but repeated three times and the estimates are averaged over the three attempts.

James [6] uses a synthetic model to generate 100 and 1000 data sets of a (unknown) fixed size and a large (unknown sized) test set in order to obtain bias and variance estimates. For UCI data sets, 5-fold cross-validation was used and from the non-test set 50 bootstrap samples were drawn. Bias and variance were averaged over the folds. One UCI data set comes with a separate test set, and for this one no cross validation was used but just a single run was done.

Valentini and Dietterich [7] use synthetic data sources to generate 100 training sets of 200 and 400 instances and a test set of 10.000 instances. Also, real data was split into data source and test set where 200 data sets of 100 instances are drawn with replacement from the data source.

Valentini and Dietterich [8] select 200 bootstrap samples (with replacement) from a data set, trains algorithms on the data set and applies the classifier to samples not selected for the training set (the so called 'out of bag' set). For

each instance, a bias and variance is estimated and the final bias and variance estimates are obtained by averaging over the instances.

Webb [9] uses 10 times repeated 3 fold cross validation and averages results over runs and folds. From each train set source, 100 samples are drawn for training sets.

Webb and Conilione [10] use repeated (10 and 50 times) fold cross validation where the training set source is sampled from the non-test set. Parameters for training set size and training set dependency can be provided for the training sets that are sampled from the training source.

3 Problems Identified

It appears that every researcher uses her own favorite method for doing bias variance measurement experiments, which makes comparison among various works hard. Also, it appears that ad-hoc methods are used when the experiments do not behave well. In particular when estimates turn out to be insufficiently stable the number of data sets is arbitrary increased till the estimates appear stable.

The following concrete problems can be identified;

Gold standard. Synthetic data sources can be created so that beforehand the systematic error (noise, Bayesian rate) can be determined. However, it is not clear how the type of data source impacts on the bias and variance of variance. For example, the LED data generator [11] which can be interpreted as a naive Bayes model for generating data sets may result in more stable estimates than when a decision tree is used as data source. Also, the number of attributes, amount of noise in the attributes, cardinality of nominal attributes, presence of numerical attributes and other data source characteristics may influence experimental results.

Empirical standard. No generally accepted protocol exists for measuring bias and variance for a given data set. The main features in which methods differ are the following;

- Variants for obtaining data sources are train/test split of data, cross validation and bootstrap.
- The way data source are used to create data sets using sampling with or without replacement. Sampling with replacement results in data sets with possibly different characteristics than the original data set. Also, it impacts on some learning algorithms, for instance, 1-nearest neighbor.
  Another way data source use varies is that some instances can be ignored in order to control the interdependency between sampled data sets.
- The number of data sets sampled from each data source, which ranges from 50 to 1000 in the literature.
- The number of times the process is repeated to generate more stable estimates, e.g. once or 10 times repeated cross validation.

Desiderata. Most articles do not explicitly list the properties of a bias variance estimator that are desirable. In this article, we concentrate on the following criteria;

- *Unbiased estimates.* Just as for any estimator.
- *Stability/replicability.* Repeated measurements on the same data with different random splits of this data should result in (almost) equal estimates.
- *Efficiency.* As little computational effort should be required for estimating bias and variance.
- *Data source control.* Most bias variance measurements are applied to a particular 'real world' data set of size n but no method allows for sampling data sets of size n (without duplicating data). The size of the data sets is one item to control. Another is dependency between data sets.

4 Simulation Study

To examine the parameters mentioned in the previous section, we performed a number of experiments. We considered a range of data sources.

Bayesian Network Data Sources. For data sources, Bayesian networks with increasing numbers of complexity were randomly generated. Figure 3 shows the network structures and marginal distributions. The first has the same topology as Naive Bayes, the second is a tree augmented naive Bayes (TAN) structure and the third is a general Bayesian network structure, each structure representing increasingly complex concepts. The networks have 10 binary variables each and 50%/50% class probabilities.

Agrawal Data Sources. Agrawal et al. [12] defined a set of ten functions for machine learning benchmarking. These are functions over the attributes salary, commission, age, education level, car, zipcode, house value, years house owned and total loan amount. The functions are classification functions splitting the population into two groups and can be as simple as splitting on age in the interval 40 to 60 years. Others are more complex functions like testing whether $0.67(salary+commission)-5000\cdot eduction\,level+0.2\cdot equity-20.000$ is positive where *equity* is a hidden variable calculated as 0 if the house is owned less than 20 years or $0.1 \times house\ value \times (years\ house\ owned - 20)$ otherwise.

Gaussian Radial Base Data Sources. By creating a random set of centers with randomly assigned weights, a set of random Gaussian radial base functions can be defined around those centers. By selecting a center at random according to the weights of the centers and generating attributes randomly offset from the center and assigning the class attribute associated with the center, new instances can be generated. This type of data generator is called the random RBF generator. In the experiments, random RBF data generator from Weka [2] were used with 10, 20, 50 and 100 centers.

Number of Samples. To eliminate dependence between samples on bias/variance estimates, initially we used the data sources to produce new independent samples every time a sample is required. After determining other parameters, sampling methods (resampling, cross validation and bootstrapping) based on single data sets are considered.

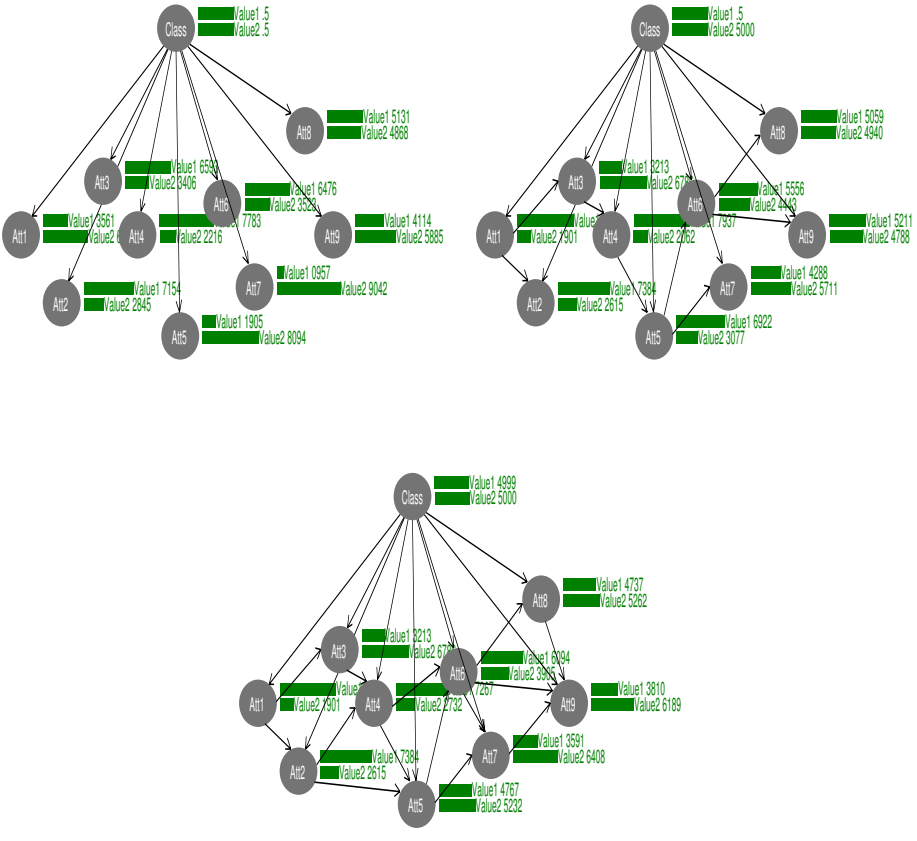


Fig. 3. Bayesian networks used as data generators

The first experiment aims at determining the number of samples required for getting a stable estimate of the bias and variance. Also, it tries to get an impression on the impact of the data source on the variability of bias/variance estimates. Up to 2000 samples were drawn from the Bayesian networks and C45 as implemented in Weka [2] was trained and then tested on a set of 10.000 instances. This process was repeated 100 times for the three networks, so 60.000 trees were trained in this experiment. Figure 4 shows the results.

The same experiment as for the naive Bayes, TAN and Bayesian net data generator was repeated for the ten Agrawal functions (for which a total of 10x100x2000= 200.000 trees were learned) and with the random RBF data generators (for which a total of 4x100x2000= 80.000 trees were learned). However, results are not shown due to space limitations.

In general, the experiments show that taking less than 200 samples tends to have a significant impact on the variability of the bias/variance estimates. This implies most results published in the literature (as outlined in Section 2.1) can be expected to suffer from unstable estimates. The variability in the

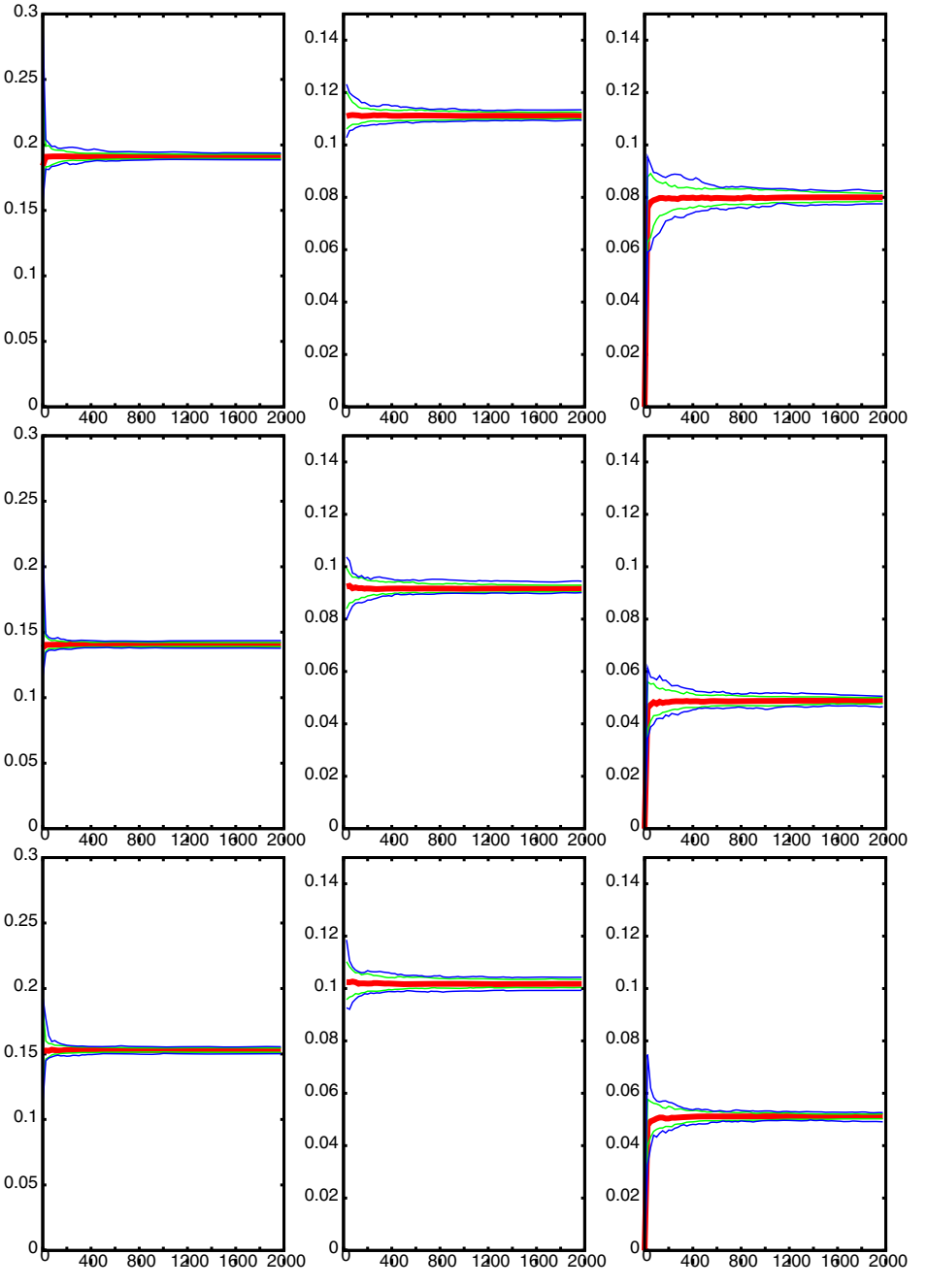


Fig. 4. Comparing different complexity of concepts on bias/variance for C4.5; naive Bayes data source in top row, TAN in middle row and full Bayesian net in bottom row. Error (left), bias (middle) and variance (right) estimates and their 100% and 90% bounds on the y-axis. Number of datasets sampled on the x-axis (ranging 0 to 2000).

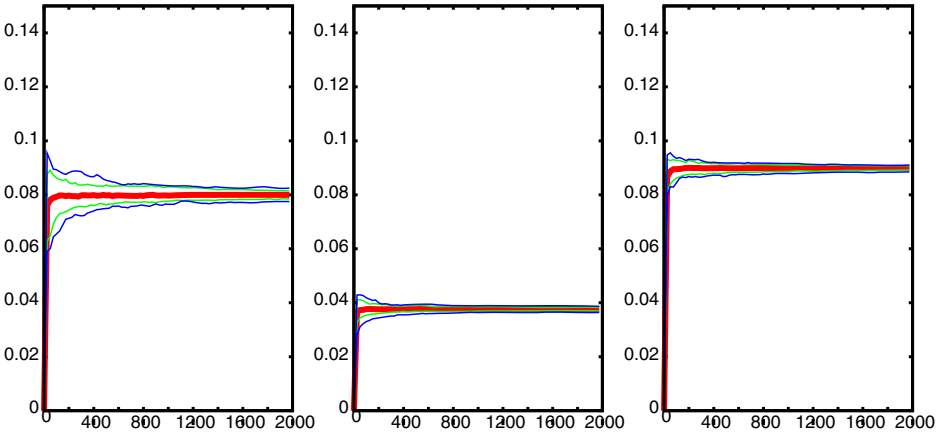


Fig. 5. Comparing various learning algorithms; left C4.5, middle Naive Bayes, right nearest neighbor. Variance estimates and their 100% and 90% bounds on the y-axis. Number of datasets sampled on the x-axis (ranging 0 to 2000).

estimates does not decrease very much when taking more than 1000 samples. The variability of the estimates for large number of samples differs with every data source. However, as Figure 4 shows, the variability does not necessarily increase with increasingly complex data sources. The above observations hold for the ten Agrawal data sources and random RBF generators as well.

Learning algorithm. To get an impression how sensitive the bias variance decomposition is to the actual learning algorithm under investigation, we reran the above experiment with C4.5, naive Bayes and nearest neighbor as learning algorithms. Figure 5 shows a result typical for the outcomes for the naive Bayes data source (remainder not shown due to limited space). Clearly, different algorithms have different bias and variance, but also the variability of the estimates differs with different algorithms. Stable algorithms like naive Bayes appear to result in less variability than highly unstable ones like C4.5 [13]. Nearest neighbor is a medium stable algorithm and has variability of estimates in between the other two algorithms. The same observations hold for the other data sources.

Test Set Size. The test set size can be expected to have an impact on the bias variance decomposition since small test sets can be expected to result in highly variable decompositions. Likewise, virtual infinite test set sizes can be expected to result in more stable decompositions. To get a sense where the desired balance between stable decompositions and acceptable computational effort lies, we ran the experiment with different test set sizes and Figure 6 shows the results. Clearly, test set sizes under 2000 result in highly variable decompositions. Improvement in the stability of the decomposition vanishes for test sets over 10.000 instances.

Sampling Method. The sampling method was reported to have some effect on the variability of the bias variance estimates [10], in particular cross

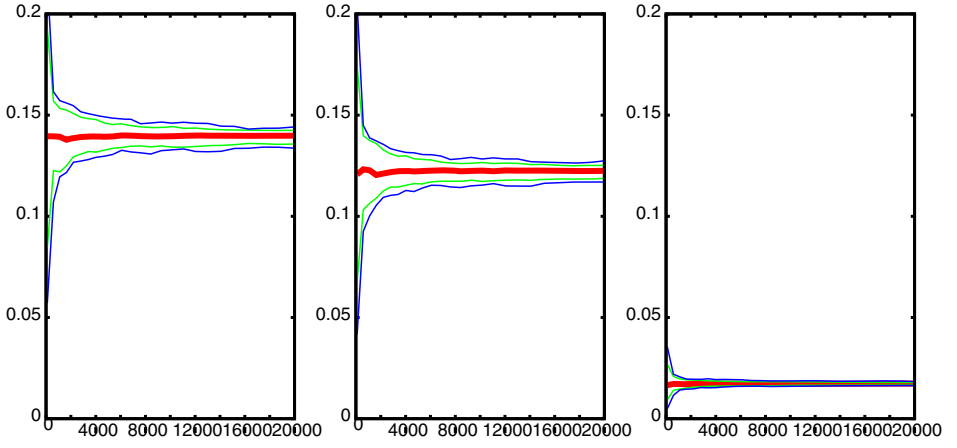


Fig. 6. Effect of test set size on stability of error/bias/variance estimates. On x-axis the test set size from 100 to 20.000 instances. Naive Bayes data source. On y-axis the error (left plate), bias (middle) and variance (right) with median, 90% interval and 100% interval over 100 runs.

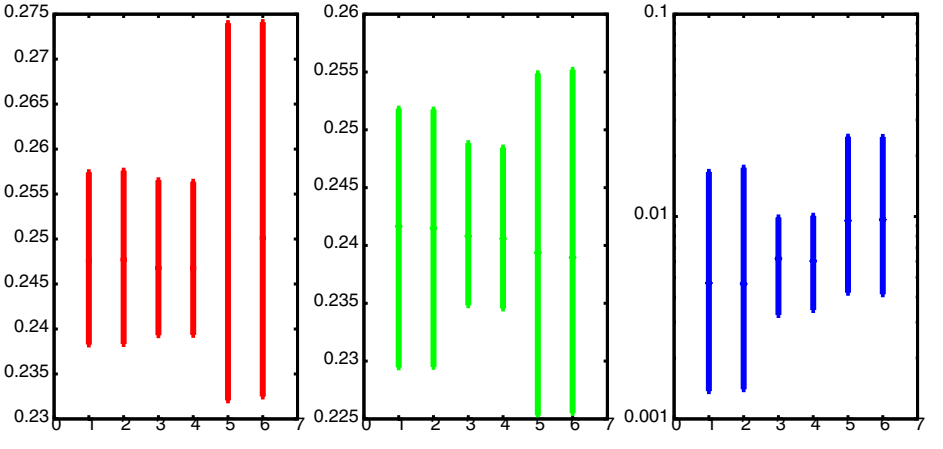


Fig. 7. Mean error (left), bias (middle) and variance (right) with 90% interval for 1000× and 10000× resampling, 1× and 10× 10 fold cross validation and 1000× and 10000× bootstrapping respectively. Note linear y-scale for error and bias and log scale for variance.

validation appeared to result in more stable outcomes than sampling with replacement. Another approach is bootstrapping with out of bag instance classification for estimates [8]. Figure 7 shows the mean and 90% interval for error (left plate), bias (middle) and variance (right) estimates using sampling with replacement (first pair of items in plate), cross validation (next pair of items) and

bootstrapping (last pair). The estimates are for C4.5 on the naive Bayes data source from Section 4 and the training sets contain 1000 instances. Computationally, resampling is cheapest, cross validation takes a bit more due to some extra administration and bootstrapping takes even more due to the administration that comes with out of bag estimation. The first item in each pair is where 1000 samples are taken from the data source, and the second where 10.000 samples were taken. For cross validation, ten folds were used and within each fold 100 samples were taken. To get the 10.000 samples the process was repeated ten times and estimates averaged over the ten runs. Figure 7 shows clearly that the variability for cross validation is considerably less than that for resampling, confirming [10]. Furthermore, it appears that bootstrapping results in higher variability of bias/variance estimates. Furthermore, increasing the number of samples has little effect on the variability of the estimates, so this variability is inherent in the samplings method.

5 Discussion/Conclusions

We identified the following issues that have an impact on the stability of bias variance decompositions

- number of samples drawn from a datasource. Decompositions using less than 200 samples result in highly unstable estimates. This is surprising since proposals found in the literature routinely use 100 or less samples. Consequently, it is easy to draw erroneous conclusions from such simulations (as illustrated by Figure 1).
- learning algorithm. Unstable algorithm like C4.5 can result in twice the variability of a decomposition as stable algorithms like naive Bayes. Increasing the number of samples can reduce this effect somewhat, but cannot totally eliminate the variability due to instability of the learning algorithm.
- test set sizes under 2000 result in highly variable decompositions and are not recommended. Test set sizes of 10.000 and over do not seem to reduce variability any more. So, as a rule, the larger the test set size the lower the variability.
- sampling algorithm. Cross validation gives the least variable estimates, bootstrapping the most and resampling goes in between.
- The data source has a small effect on the variability, but no pattern could be found to determine when it can be justified to use fewer samples.

Based on these observations, we recommend to use ten fold cross validation as sampling method and take 100 samples within each fold with a test set size of at least 2000. Taking fewer samples or using a lower number of folds such that the number of samples (i.e. the number of folds times number of samples per fold) is at least 200 may be justified if the learning algorithm under consideration is very stable.

Acknowledgements

I thank Geoff Holmes for the helpful comments on improving the paper.

References

1. Kohavi, R., Wolpert, D.H.: Bias plus variance decomposition for zero-one loss functions. In: Saitta, L. (ed.) Machine Learning: Proceedings of the Thirteenth International Conference, pp. 275–283. Morgan Kaufmann, San Francisco (1996)
2. Witten, I., Frank, E.: Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco (2000)
3. Kong, E.B., Dietterich, T.G.: Error-correcting output coding corrects bias and variance. In: Proceedings of the 12th International Conference on Machine Learning, pp. 313–321. Morgan Kaufmann, San Francisco (1995)
4. Domingos, P.: A unified bias-variance decomposition and its applications. In: International Conference on Machine Learning, ICML 2000, pp. 231–238 (2000)
5. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36(1-2), 105–139 (1999)
6. James, G.: Variance and bias for general loss functions. Machine Learning 51, 115–135 (2003)
7. Valentini, G., Dietterich, T.G.: Bias-variance analysis and ensembles of svm. In: Multiple Classifier Systems: Third International Workshop, pp. 222–231 (2002)
8. Valentini, G., Dietterich, T.G.: Low bias bagged support vector machines. In: International Conference on Machine Learning, ICML 2003 (2003)
9. Webb, G.I.: Multiboosting: A technique for combining boosting and wagging. Machine Learning 40(2), 159–196 (2000)
10. Webb, G.I., Conilione, P.: Estimating bias and variance from data (unpublished manuscript) (2002), `http://www.csse.monash.edu.au/~webb/files/webbconilione06.pdf`
11. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees, pp. 43–49. Wadsworth International Group, Belmont (1984)
12. Agrawal, R., Imielinski, T., Swami, A.: Database mining: A performance perspective. IEEE Transactions on Knowledge and Data Engineering 5(6), 914–925 (1993); Special issue on Learning and Discovery in Knowledge-Based Databases.
13. Dwyer, K., Holte, R.C.: Decision tree instability and active learning. In: Kok, J.N., Koronacki, J., de Mántaras, R.L., Matwin, S., Mladenic, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 128–139. Springer, Heidelberg (2007)

Using Gaussian Processes to Optimize Expensive Functions

Marcus Frean and Phillip Boyle

Victoria University of Wellington, P.O. Box 600,
Wellington, New Zealand
marcus@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/∼ marcus

Abstract. The task of finding the optimum of some function $f(\mathbf{x})$ is commonly accomplished by generating and testing sample solutions iteratively, choosing each new sample $\mathbf{x}$ heuristically on the basis of results to date. We use Gaussian processes to represent predictions and uncertainty about the true function, and describe how to use these predictions to choose where to take each new sample in an optimal way. By doing this we were able to solve a difficult optimization problem - finding weights in a neural network controller to simultaneously balance two vertical poles - using an order of magnitude fewer samples than reported elsewhere.

1 Introduction

One potentially efficient way to perform optimisation is to use the data collected so far to build a predictive model, and use that model to select subsequent search points. In an optimisation context, this model is often referred to as a *response surface*. This method is potentially efficient if data collection is expensive relative to the cost of building and searching a response surface. In many such cases, it can be beneficial to use relatively cheap computing resources to build and search a response surface, rather than incur large costs by directly searching in the problem space. A case in point is the construction of robotic control systems.

In the response surface methodology [1] we construct a response surface and search that surface for likely candidate points, measured according to some criterion. Jones [2] provides a summary of many such methods and discusses their relative merits. As a simple example, consider a noiseless optimisation problem where, given an intial set of samples, we proceed as follows:

1. Fit a basis function model to the data.
2. Find an optimum point of the model and call this point $\mathbf{x}_{\text{new}}$
3. Sample the problem at $\mathbf{x}_{\text{new}}$, and add the result to the current data set.
4. Repeat until satisfied or until is no satisfactory progress is being made.

This is a poor general purpose optimisation algorithm, and for several reasons. One symptom is that it rapidly becomes stuck in one region of the search space, wasting further samples there despite already having good information about it from previous samples.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 258–267, 2008.

A more sophisticated search method might attempt to capture regularities about the nature of the search space (rather than merely fitting the existing data), and then use that model more sensibly than simply suggesting the highest predicted point for the next sample. The tendency to *explore* unchartered territory and collect new information about the problem's structure once local territory has been mapped should be an emergent property of a good search algorithm, not a heuristic to be wired in as a quick fix for "premature convergence". This naturally leads us to consider statistical models, where we have a full predictive distribution rather than a single prediction at each search point. Gaussian process models [16] are attractive from this standpoint, for 3 reasons: (i) the predictive distribution is easily obtained, (ii) a Bayesian treatment of hyperparameters allows the model to learn general properties of the surface (smoothness etc.) from previous samples, and (iii) a sensible criterion for drawing the next sample is easily obtained from them.

2 Gaussian Processes

Given training data $\mathcal{D}$ consisting of N "input" vectors $\mathbf{x}_i$ paired with scalar "outputs" y_i for $i \in \{1, 2, ..., N\}$, Gaussian process regression is a machine learning technique for infering likely values of y for a novel input $\mathbf{x}$. The study of Gaussian processes for prediction began in geostatistics with kriging [3], [4] and O'Hagan's [5] application to one-dimensional curve fitting. Buntine [6], MacKay [7], and Neal [8] introduced a Bayesian interpretation that provided a consistent method for handling network complexity (see [9,10] for reviews), followed by regression in a machine learning context [11,12,13]. See [14,15,16] for good introductions. Interesting machine learning applications include reinforcement learning [17], incorporation of derivative observations [18], speeding up the evaluation of Bayesian integrals [19,20], and as models of dynamical systems [21].

The key assumption is that the posterior distribution $p(y|\mathbf{x}, \mathcal{D})$ is Gaussian. To compute its mean and variance, one specifies a valid covariance function $\mathrm{cov}(\mathbf{x}, \mathbf{x}')$, and defines vector $\mathbf{k}$ where $k_i = \mathrm{cov}(\mathbf{x}, \mathbf{x}_i)$, and matrix $\mathbf{C}$ where $C_{ij} = \mathrm{cov}(\mathbf{x}_i, \mathbf{x}_j)$. A common choice of covariance function is the squared exponential, with a length scale r_d associated with each axis:

$$\mathrm{cov}(\mathbf{x}_i, \mathbf{x}_j) \;=\; \alpha \exp\left[-\frac{1}{2} \sum_{d=1}^{D} \frac{(x_i^{(d)} - x_j^{(d)})^2}{2 r_d^2} \right] \;+\; \beta \delta_{ij} \qquad (1)$$

Here $\boldsymbol{\theta} = \{\alpha, r_1, \ldots, r_D, \beta\}$ are hyperparameters, for which Maximum a posteriori (MAP) values can be inferred from the data [16] by maximising the posterior density $p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y})$, which is the product of the likelihood function and a prior density over hyperparameters. In the experiments here we used the following log normal priors for $p(\boldsymbol{\theta})$: $\log \alpha \sim \mathcal{N}(-\log 0.5, 0.5)$, $\log r_d \sim \mathcal{N}(-\log 0.5, 0.5)$, and $\log \beta \sim \mathcal{N}(-\log 0.05, 0.5)$.

At any test point $\mathbf{x}$ we then have a predictive distribution that can be shown to be Gaussian with mean $y(\mathbf{x}) = \mathbf{k}^{\mathrm{T}} \mathbf{C}^{-1} \mathbf{y}$, and variance $s^2(\mathbf{x}) = \kappa - \mathbf{k}^{\mathrm{T}} \mathbf{C}^{-1} \mathbf{k}$ where $\kappa = \mathrm{cov}(\mathbf{x}, \mathbf{x})$ (*eg.* for the above covariance function $\kappa = \alpha + \beta$).

3 Expected Improvement

If the model we build provides a predictive distribution at any test point, we can use it to ask what improvement, over our current best sample, do we expect to get from sampling at any test point. Such a measure is known as the expected improvement (e.g. see [2]). The expected improvement ("EI") is particularly straightforward to calculate in the case of a Gaussian process.

For a maximisation problem, the predicted improvement at $\mathbf{x}$ is $I(\mathbf{x}) = \hat{y}(\mathbf{x}) - f_{\text{best}}$, where f_{best} is the current best score and $\hat{y}(\mathbf{x})$ is the model's prediction at $\mathbf{x}$. The prediction is Gaussian distributed as $\hat{y}(\mathbf{x}) \sim \mathcal{N}(y(\mathbf{x}), s^2(\mathbf{x}))$, and so is the improvement: $I \sim \mathcal{N}(y(\mathbf{x}) - f_{\text{best}}, s^2(\mathbf{x}))$. The expected improvement at $\mathbf{x}$ for models with Gaussian predictive distributions is therefore

$$EI(\mathbf{x}) \;= E[\max\{0, I(\mathbf{x})\}] \;=\; \int_{I=0}^{I=\infty} Ip(I)dI \;=\; s(\mathbf{x})\,[u\,\Phi(u) + \phi(u)]$$

where $u = \frac{y(\mathbf{x}) - f_{\text{best}}}{s(\mathbf{x})}$. The functions $\Phi(\cdot)$ and $\phi(\cdot)$ are the normal cumulative distribution and normal density function respectively:

$$\Phi(u) = \frac{1}{2}\,\text{erf}\left(\frac{u}{\sqrt{2}}\right) + \frac{1}{2} \qquad\qquad \phi(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$$

Figure 1 illustrates the concept of expected improvement for a GP model in a maximisation context.

To find a new search point that maximises the expected improvement, we can also make use of gradient information. The gradient of EI with respect to $\mathbf{x}$ is:

$$\frac{\partial EI(\mathbf{x})}{\partial \mathbf{x}} \;=\; \Big[u\Phi(u) + \phi(u)\Big]\frac{\partial s(\mathbf{x})}{\partial \mathbf{x}} \;+\; s(\mathbf{x})\Phi(u)\frac{\partial u}{\partial \mathbf{x}}$$

(the other two terms cancel) where

$$\frac{\partial s(\mathbf{x})}{\partial \mathbf{x}} = -\left(\frac{\partial \mathbf{k}^{\mathrm{T}}}{\partial \mathbf{x}}\mathbf{C}^{-1}\mathbf{k}\right)\Big/ s(\mathbf{x}), \quad \text{and} \quad \frac{\partial u}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{k}^{\mathrm{T}}}{\partial \mathbf{x}}\mathbf{C}^{-1}\mathbf{y} - u\frac{\partial s(\mathbf{x})}{\partial \mathbf{x}}\right)\Big/ s(\mathbf{x})$$

The Jacobian $\frac{\partial \mathbf{k}^{\mathrm{T}}}{\partial \mathbf{x}}$ is dependent on the form of the covariance function: it is $D \times N$ matrix whose $(i,j)^{\text{th}}$ element is $\frac{\partial \text{cov}(\mathbf{x}, \mathbf{x}_j)}{\partial x_i}$ where $\mathbf{x} = [x_1 \ldots x_D]^{\mathrm{T}}$.

4 GPO

In this section, the Gaussian Processes for Optimisation (GPO) algorithm is described. Our goal is find the position $\mathbf{x}_{\text{opt}}$ of the optimum of a surface $f^*(\cdot)$, using as few samples as possible, so we update *a*ll parameters in the model as each new data point arrives. GPO begins by assuming a start point $\mathbf{x}_0$, which in general is sampled from some distribution reflecting our prior beliefs about $f^*(\cdot)$. At each iteration, the algorithm builds a GP model of the current data by finding

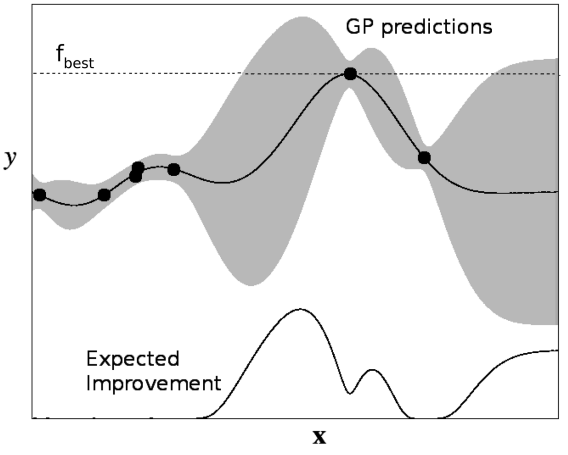


Fig. 1. Expected Improvement for a GP model in a maximisation context. Data points are shown as black dots. The GP model's predictive distribution has a mean shown by the black line that fits the data, and a standard deviation shown by the shaded region surrounding the mean. The black line at the bottom shows the expected improvement given this particular GP model. Notice that expected improvement is maximum when the model's prediction is better than or close to f_{best} *and* the predictive variance is high. Expected improvement is quite high when when the model prediction is low *and* the predictive variance is high. But when the model prediction is low *and* the predictive variance is low, the expected improvement is almost zero.

$\boldsymbol{\theta}_{MAP}$. If multimodal posterior distributions are considered a problem, then the GP model can be built by restarting the log posterior density maximisation multiple times from samples drawn from $p(\boldsymbol{\theta})$.

The resulting GP model is used to select the next $\mathbf{x}_{\text{new}}$ to evaluate by finding a point that maximises the expected improvement. This can be achieved by using the gradient of the expected improvement as input to (*eg.*) the conjugate gradient algorithm [23]. To overcome the problem of suboptimal local maxima, multiple restarts are made starting from randomly selected points in the current data set $\mathbf{X}$. The new observation y_{new} is found from $f^*(\mathbf{x}_{\text{new}})$ and the results are added to the current data set. Iterations continue until some stopping criterion is met.

Figure 2 shows the results of running standard GPO on a simple $1D$ toy problem. Notice how the search initially focuses on the suboptimal maximum on the left. However, once the algorithm has sampled here a few times, the expected improvement of sampling in this suboptimal region diminishes quickly. At this point, the search expands into new regions. The algorithm will exploit local knowledge to make improvement, but will also explore when the expected returns of this exploitation decrease.

Notice that the squared exponential form of the covariance function is "axis-aligned" in the sense that it assigns a separate length scale to each direction in

Algorithm 1. GPO

Input: optimisation problem $f^*(\cdot)$, and starting point $\mathbf{x}_0$

$\mathbf{x}_{\text{best}} \leftarrow \mathbf{x}_0$, $y_{\text{best}} \leftarrow f^*(\mathbf{x}_0)$;
$\mathbf{y} \leftarrow [y_{\text{best}}]$, $\mathbf{X} \leftarrow \mathbf{x}_{\text{best}}$;
repeat
  $\boldsymbol{\theta}_{MAP} \longleftarrow \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y})$;
  $\mathbf{x}_{\text{new}} \longleftarrow \arg\max_{x} EI(\mathbf{x}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}_{MAP})$;
  $y_{\text{new}} \leftarrow f^*(\mathbf{x}_{new})$;
  if $y_{new} \geq y_{best}$ **then**
    $y_{\text{best}} \leftarrow y_{\text{new}}$, $\mathbf{x}_{\text{best}} \leftarrow \mathbf{x}_{\text{new}}$;
  $\mathbf{X} \leftarrow [\mathbf{X} \,|\, \mathbf{x}_{\text{new}}]$, $\mathbf{y} \leftarrow \left[\mathbf{y}^{\mathrm{T}} \,|\, y_{\text{new}}\right]^{\mathrm{T}}$;
  $\mathbf{V}_{rot} \leftarrow \arg\max_{\mathbf{V}} p(\boldsymbol{\theta}^{V}_{MAP}|\mathbf{VX}, \mathbf{y})$ where $\boldsymbol{\theta}^{V}_{MAP} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{VX}, \mathbf{y})$;
  $\mathbf{X} \leftarrow \mathbf{V}_{rot}\mathbf{X}$
until *(stopping criteria satisfied)*;
return $\mathbf{x}_{best}$

input space. However in optimization problems in more than one dimension this is a poor assumption, since there will often be interactions between input variables which induce covariance structure that is "off-axis". To allow the algorithm to discover such structure, at each step we test out a number of random rotations $\mathbf{X}' = \mathbf{V}_{rot}\mathbf{X}$ of the current input data $\mathbf{X}$ using randomly generated orthonormal matrices $\mathbf{V}_{rot}$, and choose the rotated data set for which $p(\boldsymbol{\theta}_{MAP}|\mathbf{X}', \mathbf{y})$ is greatest. This preprocessing of the data is equivalent to learning a rotated covariance function and allows GPO deal with off-axis structure in the properties of the search surface. Further details are given in [20].

Jones [2] first introduced kriging for optimisation using expected improvement to select the next iterate. Büche, Schraudolph and Koumoutsakos [22] explicitly used Gaussian processes for optimisation, and demonstrated the algorithm's effectiveness on a number of benchmark problems. This work did not make use of expected improvement, did not place prior distributions over the hyperparameters, and did not consider the deficiencies of using an axis-aligned covariance function to optimise objective functions with correlated output (dependent) variables. The algorithm presented here takes all these factors into account. Recently [28] have used similar ideas to those presented here to optimize the gait of a mobile robot, although they use a different criterion (probability of *any* improvement) and don't deal with correlated variables.

5 Double Pole Balancing with GPO

The double pole balancing task consists of two upright poles (or inverted pendulums), attached by hinges to a cart. The goal is to keep the two poles balanced by applying a $[-10, 10]N$ force to the cart. Balanced poles are defined as within $\pm 36\,^\circ$ from vertical, and the cart is limited to a track which is $4.8m$ long. The

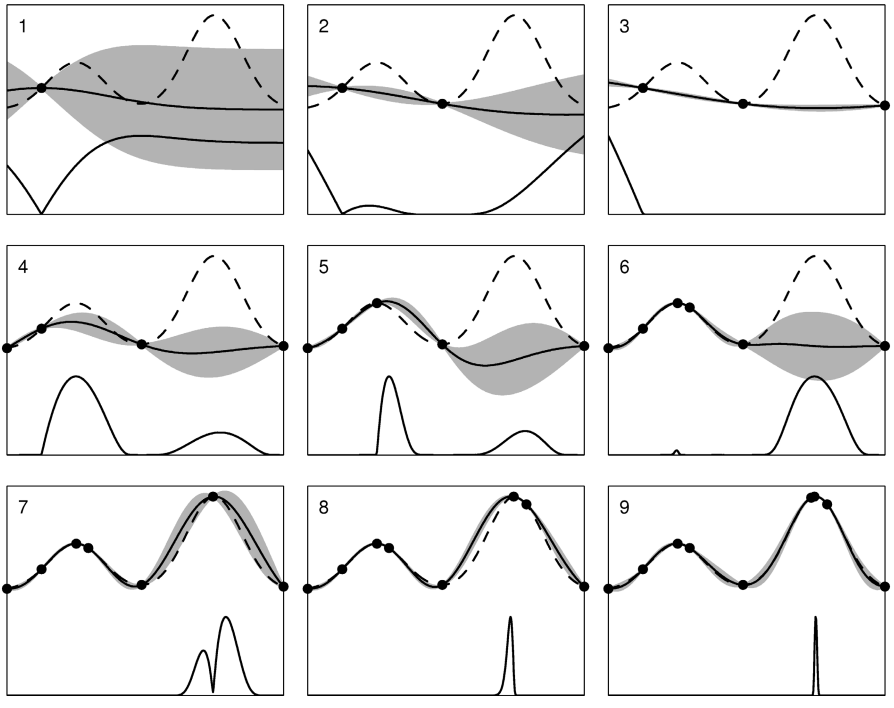


Fig. 2. GPO applied to a $1D$ function for 9 iterations. The dashed line shows the underlying function to be optimised. The sample points are shown as black dots, along with the model's predictive distribution (the line surrounded by the shaded area shows the mean and standard deviation). The expected improvement is rescaled and shown by the solid line at the bottom of each window. Note the hill-climbing behaviour (*eg.* iterations 4-5) exploiting regularity of the surface, and exploratory behaviour at other times (*eg.* iteration 6).

controller is supplied with inputs from sensors measuring the cart's position and velocity $x, \dot{x}$ and the angle and angular velocity of each pole with respect to the cart $\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2$. The poles have different lengths and masses (pole 1 is $0.1m$ and $0.01kg$; pole 2 is $1.0m$ and $0.1kg$) and the system is noiseless with initial state vector $\mathbf{s} = [x\ \dot{x}\ \theta_1\ \dot{\theta}_1\ \theta_2\ \dot{\theta}_2]^{\mathrm{T}} = [0\ 0\ 0\ 0\ \frac{\pi}{180}\ 0]^{\mathrm{T}}$, where angles are measured in rad from vertical, and angular velocities are measured in radians / sec. The centre of the track is defined as $x = 0$, and is the position of the cart at the beginning of the task. Note that this task is Markovian as the full system state vector $\mathbf{s}$ is available to the controller and is the same as the "double pole balancing with velocity information" problem as presented by Stanley and Miikkulainen [24,25,26].

If the goal is to keep the poles balanced for as long as possible, one solution is to wiggle the poles back and forth about a central position. To prevent this,

Gruau [27] defined a fitness function that penalises such solutions, $f_{\text{gruau}} = 0.1f_1 + 0.9f_2$, [24,26]. The two components are defined over 1000 time steps (10 seconds simulated time):

$$f_1 = t/1000 \tag{2}$$

$$f_2 = \begin{cases} 0 & \text{if } t < 100, \\ \frac{0.75}{\sum_{i=t-100}^{t}(|x^i|+|\dot{x}^i|+|\theta_1|+|\dot{\theta}_1|)} & \text{otherwise.} \end{cases} \tag{3}$$

where t is the number of time steps both poles remained balanced during a trial lasting 10 seconds. f_{gruau} can be maximised by keeping both poles balanced, and by maintaining the cart steady at the centre of the track during the final second of the trial. Effectively, to maximise f_{gruau} the controller must balance the poles without 'wiggling'. As the denominator of (3) approaches zero f_2 approaches infinity, so f_{gruau} was non-linearly rescaled into the range $[0, 1]$ giving $f_{\text{gpo}} = \tanh(f_{\text{gruau}}/2)$. Controllers were considered successful solutions when $f_{\text{gpo}} \geq \tanh\left(\frac{5}{2}\right)$.

5.1 Feedforward Neural Network Controllers

The double pole balancing task described above is a non-linear, unstable control problem. However, because the poles have different lengths and masses, the system is controllable. In addition, the task is Markovian. Overall, full knowledge of the system state is sufficient to balance the poles, and this can be achieved with a mapping from $\mathbf{s}$ to u, our control force. In other words, there exists at least one mapping $\mathbf{s} \mapsto u$ that is capable of balancing the poles. A successful controller must functionally approximate such a mapping.

The control force is implemented by a feedforward neural network with a single hidden layer having H units, and output limited to $[-10, 10]N$:

$$u = 10 \tanh\left(\mathbf{w}_o^{\mathrm{T}} \tanh\left(\mathbf{W}_i^{\mathrm{T}}\mathbf{s} + \mathbf{b}\right)\right)$$

where $\mathbf{w}_o$ is an $H \times 1$ vector of output weights, $\mathbf{b}$ is an $H \times 1$ vector of biases, $\mathbf{W}_i$ is a $6 \times H$ matrix of input weights, $\mathbf{s}$ is the 6×1 state vector.

5.2 Optimisation and Incremental Network Growth

We optimized $f^\star = f_{\text{gruau}}(\mathbf{w}_o, \mathbf{W}, \mathbf{b})$. The optimisation started with a single unit in the network, $H = 1$. Initially, therefore, there were 8 parameters that need optimising. GPO, with the axis-aligned covariance function (Eq. 1) and data rotation prior to training, was used optimise these weights until either there had been no improvement in the best fitness for 64 consecutive samples, or 250 samples had been taken. When either of these conditions were met, the current optimised parameters were frozen, and a new unit with zeroed weights

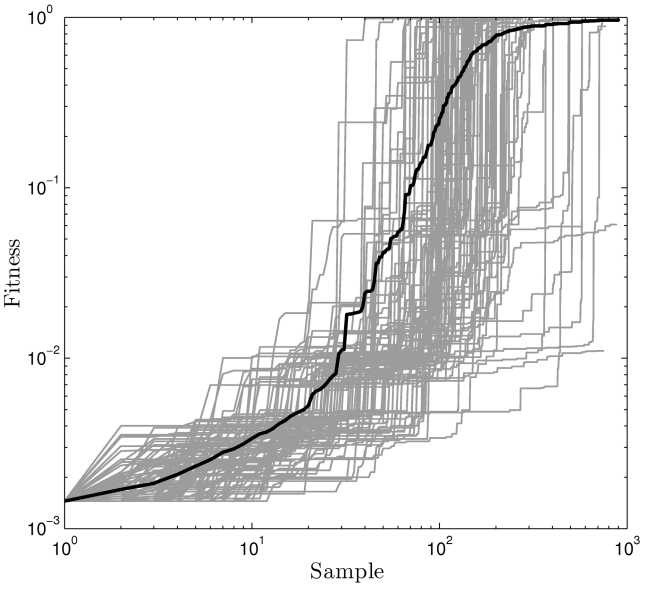


Fig. 3. Double Pole Balancing with Gruau fitness, optimised using GPO. The figure shows 100 separate optimisations (grey) and the average (black).

was added to the network. The cycle repeated until a solution was found, or 5 units and their weights had been optimised. Note that the initial weights for the first iteration were zero (the algorithm started from the same place every time it was run), and 8 parameters were being optimised at every stage.

Figure 3 shows 100 runs of GPO on this task. 96 of these runs found a successful controller solution with $f_{\text{gruau}} \geq \tanh\left(\frac{5}{2}\right)$ in < 1000 evaluations (samples). The median number of evaluations required to find a successful controller was 151, and the mean was 194. The majority (78%) of successful controllers used only 1 unit in their solution (i.e. 6 input weights, 1 bias and 1 output weight). 12 runs used 2 units, while 9 needed from 3 to 5 units.

Stanley and Miikkulainen [24,25,26] introduced "Neuroevolution of Augmenting Topologies" (NEAT), and applied it to a number of pole balancing tasks, including the double pole balancing problem presented above. The NEAT method is a genetic algorithm with mutation and crossover operations specially crafted to enhance the evolution of neural network controllers. The details of NEAT are not important here, other than that it produced impressive results in solving the double pole balancing task with velocity information. NEAT required an average 3578 network evaluations to find a controller solution, which compared favourably with other results from literature.

GPO produced successful controllers in 96 out of 100 trials, and did so with a mean of 194 evaluations. This is a significant improvement over NEAT.

6 Summary

We have presented an optimization algorithm that uses Gaussian process regression to suggest where to take samples, with the goal of finding a good solution in a small number of function evaluations. GPO uses data rotation to allow for interactions between input variables, and learns about the search space as it proceeds, by finding maximum a posteriori values for hyperparameters at every step. In this way search is carried out as much as possible on the model instead of the real world, using the conjugate gradient method to find points having the highest expected improvement. Sequences of samples taken in this way exhibit a variety of intuitively sensible yet emergent properties, such as hill-climbing behaviour, and avoidance of regions the model considers to be well characterised already. Despite being a deterministic procedure, GPO also shows non-trivial exploratory behaviour, in testing out regions that seem promising under the current model.

As a demonstration of this algorithm we applied it to the task of finding optimal parameters for the double pole-balancing problem, with that result that it learns successful controllers using about 200 evaluations of possible controllers, compared to over 3500 reported for other algorithms.

References

1. Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product Optimization Using Designed Experiments, 2nd edn. Wiley-Interscience, Hoboken (2002)
2. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization 21, 345–383 (2001)
3. Matheron, G.: Principles of geostatistics. Economic Geology 58, 1246–1266 (1963)
4. Cressie, N.: Statistics for Spatial Data. Wiley, Chichester (1993)
5. O'Hagan, A.: Curve fitting and optimal design for prediction (with discussion). J. Roy. Statist. Soc. Ser. B 40, 1–42 (1978)
6. Buntine, W., Weigend, A.: Bayesian backpropagation. Complex Systems 5, 603–643 (1991)
7. MacKay, D.J.C.: Bayesian Methods for Adaptive Models. PhD thesis, California Institute of Technology (1992)
8. Neal, R.M.: Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Tech. Rep. CRG-TR-92-1, Dept. of Computer Science, Univ. of Toronto (1992)
9. MacKay, D.J.C.: Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. Network: Computation in Neural Systems 6, 469–505 (1995)
10. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
11. Williams, C.K., Rasmussen, C.E.: Gaussian processes for regression. In: Touretzsky, D., Mozer, M., Hasselmo, M. (eds.) Advances in Neural Information Processing Systems, vol. 8 (1996)

12. Rasmussen, C.E.: Evaluation of Gaussian Processes and other methods for Non-Linear Regression. PhD thesis, Graduate Department of Computer Science, University of Toronto (1996)
13. Neal, R.: Monte carlo implementation of Gaussian process models for Bayesian regression and classification. Tech. Rep. CRG-TR-97-2, Dept. of Computer Science, Univ. of Toronto (1997)
14. Gibbs, M.: Bayesian Gaussian Processes for Classification and Regression. PhD thesis, University of Cambridge, Cambridge, UK (1997)
15. MacKay, D.J.: Information theory, inference, and learning algorithms. Cambridge University Press, Cambridge (2003)
16. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge (2006)
17. Rasmussen, C.E., Kuss, M.: Gaussian processes in reinforcement learning. In: Thrun, S., Saul, L., Schlkopf, B. (eds.) Advances in Neural Information Processing Systems, NIPS (2002), vol. 16. The MIT Press, Cambridge (2002)
18. Solak, E., Murray-Smith, R., Leithead, W.E., Leith, D., Rasmussen, C.E.: Derivative observations in Gaussian process models of dynamic systems. In: Advances in Neural Information Processing Systems, NIPS (2003), vol. 15, pp. 1033–1040. The MIT Press, Cambridge (2003)
19. Rasmussen, C.E.: Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals. In: Bernardo, J.M., Bayarri, M.J., Berger, J.O., Dawid, A.P., Heckerman, D., Smith, A.F.M., West, M. (eds.) Bayesian Statistics, vol. 7, pp. 651–659. Oxford University Press, Oxford (2003)
20. Boyle, P.: Gaussian processes for regression and optimisation. PhD thesis, Victoria University of Wellington, New Zealand (2007)
21. Wang, J., Fleet, D., Hertzmann, A.: Gaussian process dynamical models. In: Weiss, Y., Schlkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems, NIPS (2006), vol. 18, pp. 1443–1450. The MIT Press, Cambridge (2006)
22. Buche, D., Schraudolph, N., Koumoutsakos, P.: Accelerating evolutionary algorithms with gaussian process fitness function models. IEEE Transactions on Systems, Man, and Cybernetics 35(2), 183–194 (2005)
23. Press, W., Flannery, B., Teukolsky, S., Vetterling, W.: Numerical recipes in C: The art of scientific computing. Cambridge University Press, Cambridge (1989)
24. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
25. Stanley, K.O., Miikkulainen, R.: Efficient reinforcement learning through evolving neural network topologies. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002). Morgan Kaufmann, San Francisco (2002)
26. Stanley, K.O.: Efficient Evolution of Neural Networks Through Complexification. PhD thesis, Department of Computer Sciences, The University of Texas at Austin (2004)
27. Gruau, F., Whitley, D., Pyeatt, L.: A comparison between cellular encoding and direct encoding for genetic neural networks. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, pp. 81–89. MIT Press, Cambridge (1996)
28. Lizotte, D., Wang, T., Bowling, M., Schuurmans, D.: Automatic gait optimization with gaussian process regression. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, pp. 944–949 (2007)

Discriminant Analysis Methods for Microarray Data Classification

Chuanliang Chen[1], Yun-Chao Gong[2], and Rongfang Bie[1,*]

[1] Department of Computer Science, Beijing Normal University, Beijing 100875, China
[2] Software Institute, Nanjing University, Nanjing 210089, China
C.L.Chen86@gmail.com, gyc05@software.nju.edu.cn,
rfbie@bnu.edu.cn

Abstract. The studies of DNA Microarray technologies have produced high-dimensional data. In order to alleviate the "curse of dimensionality" and better analyze these data, many linear and non-linear dimension reduction methods such as PCA and LLE have been widely studied. In this paper, we report our work on microarray data classification with three latest proposed discriminant analysis methods: Locality Sensitive Discriminant Analysis (LSDA), Spectral Regression Discriminant Analysis (SRDA), and Supervised Neighborhood Preserving Embedding (S-NPE). Results of experiments on four data sets show the excellent effectiveness and efficiency of SRDA.

Keywords: Discriminant analysis, Dimensionality reduction, Data mining, Bioinformatics.

1 Introduction

The invention of DNA microarrays has spurred numerous efforts to acquire relative mRNA expression information from complex cellular systems [7,8,9]. A typical microarray data set usually contains expression levels for thousands of genes across hundreds of "conditions" [9], and therefore these data have quite high dimensionality. On the other hand, the number of available samples is usually very low. This is known as the problem of "large p and small n" [10]. Given the fact that there are not many observations that are scattered in very high-dimensional space [10], dimensionality reduction is necessary, which is also crucial for reducing the complexity and improving the performances of machine learning techniques. There have been many linear and non-linear dimension reduction techniques applied for microarray data analysis: Principal Components Analysis (PCA) [11,12], Locally Linear Embedding (LLE) [13], Isometric Mapping (Isomap) [14], linear Multidimensional Scaling (MDS) [15], Linear Discriminant Analysis (LDA) [16], etc.

However, performing purely unsupervised dimension reduction techniques for classification with microarray data wastes the information provided by the labels of the training data. In this paper, we focus on studying the capabilities of discriminant

* The corresponding author.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 268–277, 2008.

analysis methods, which taking information contained by labels into account, for classification with microarray data. Three latest proposed discriminant analysis methods: Locality Sensitive Discriminant Analysis (LSDA), Spectral Regression Discriminant Analysis (SRDA), and Supervised Neighborhood Preserving Embedding (S-NPE), are firstly introduced to this aim. The remainder of this paper is organized as follows. Section 2 briefly reviews the methods used in our work. Results of our experiments and their analysis are presented in section 3. We conclude our work in section 4.

2 Methods

At first, the generic problem of linear dimensionality reduction problem is formally described as follows:

1. Given the original data $X = \{x_1, x_2, \cdots, x_n\}$ in high-dimensional space $\mathbb{R}^m$.

2. Find a matrix A that transforms the original data points into a new set of data points $Y = \{y_1, y_2, \cdots, y_n\}$ in a low-dimensional space $\mathbb{R}^d$ ($d \ll m$), such that y_i "represents" x_i, where $y_i = A^T x_i$.

During the transformation, a dimensionality reduction method attempts to retain the geometry of X as much as possible. We briefly review LDA below, and then the three novel algorithms for discriminant analysis, LSDA and SRDA, and S-NPE.

2.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) [4] attempts to maximize the linear separability between data points of different classes, and it is a supervised technique. LDA optimizes the ratio between the within-class scatter S_w and the between-class scatter S_b in the low-dimensional representation of the data, by finding a linear mapping M that maximizes the so-called Fisher criterion, that is

$$A^* = \arg\max_A \frac{tr(A^T S_b A)}{tr(A^T S_w A)} \ , \tag{1}$$

where $tr()$ denotes matrix trace, S_b and S_w are

$$S_w = \sum_c p_c \operatorname{cov}(X^c - \bar{X}^c) \ ,$$

$$S_b = \sum_c \operatorname{cov}(\bar{X}^c) = \operatorname{cov}(X - \bar{X}) - S_w \ ,$$

where p_c is the class prior of class label c, $\operatorname{cov}(X^c - \bar{X}^c)$ is the covariance matrix of the zero mean data points x_i assigned to class $c \in C$ (C is the set of classes), $\operatorname{cov}(\bar{X}^c)$ is that of the cluster means and $\operatorname{cov}(X - \bar{X})$ is the covariance matrix of the zero mean data X [5]. The optimization problem in Eq. 1 equals to solve the following generalized eigenvector problem

$$S_b\mathbf{a} = \lambda S_w\mathbf{a}$$

for the d largest eigenvectors $\mathbf{a}_i$ to form the transformation matrix A associated with their eigen-values, where $d < |C|$ since $rank(S_b) < |C|$.

2.2 Locality Sensitive Discriminant Analysis

Locality Sensitive Discriminant Analysis (LSDA) attempts to find a mapping M which maximizes the *local* margin between data points from different classes at each local area [2].

Similar to other manifold based learning methods, LSDA characterizes the local geometry of the data manifold by building nearest neighbor graphs. In LDSA, it achieves the targets of discovering both geometrical and discriminant structures of the data manifold by building two graphs: *within-class graph* G_w and *between-class graph* G_b. Let $l(x_i)$ be the label of x_i, $N_w(x_i)$ the set of neighbors sharing the same label with x_i, and $N_b(x_i)$ the set of neighbors with different labels from that of x_i. All neighbors are defined by using euclidean distance. Note that, the number of nearest neighbors is denoted by k, which has the same meaning with the k described in the subsection 2.4, and we set the k to be 3 in the experiments of this paper. For the reason why we select 3 instead of other numbers, please refer to subsection 3.1. The weighted matrices of G_w and G_b, i.e. W_w and W_b, are defined as [2]:

$$W_{b,ij} = \begin{cases} 1, & \text{if } x_i \in N_b(x_j) \text{ or } x_j \in N_b(x_i) \\ 0, & \text{otherwise.} \end{cases}$$

$$W_{w,ij} = \begin{cases} 1, & \text{if } x_i \in N_w(x_j) \text{ or } x_j \in N_w(x_i) \\ 0, & \text{otherwise.} \end{cases}$$

The optimization problem of LSDA can be reduced to finding [2]:

$$\begin{aligned} &\arg\max_{\mathbf{a}} \mathbf{a}^T X(\alpha L_b + (1+\alpha)W_w)X^T\mathbf{a} \\ &\text{s.t. } \mathbf{a}^T XD_w X^T\mathbf{a} = 1 \end{aligned} \tag{2}$$

where α is a regulative parameter and $0 \le \alpha \le 1$, D_w is a diagonal matrix whose entries are column sum of W_w, i.e. $D_{w,ii} = \sum_j W_{w,ij}$, and D_b is defined similarly, $L_b = D_b - W_b$ is the Laplacian matrix of G_b. To solve Eq. 2 equals to solve the following generalized eigenvector problem

$$X(\alpha L_b + (1-\alpha)W_w)X^T\mathbf{a} = \lambda XD_w X^T\mathbf{a} \quad .$$

Then we can transform X to d-dimensional space by selecting the top d largest eigenvectors to form A.

Since for microarray data $rank(X) \le n$ (the number of genes/features), both $X(\alpha L_b + (1-\alpha)W_w)X^T$ and $XD_w X^T$ are singular. We apply Principal Component Analysis in our experiments firstly and then remove the obtained components with zero eigenvalues [2].

2.3 Spectral Regression Discriminant Analysis

Spectral Regression Discriminant Analysis (SRDA) casts discriminant analysis into a regression framework by using spectral graph analysis [3]. It combines the spectral graph analysis and regression and is an efficient and effective discriminant analysis method [3].

In fact, SRDA is essentially originated from LDA, so for more theoretical details please refer subsection 2.1 and [3]. We give the procedure of SRDA which original described in [17].

Let data points of X belong to c classes, and m_k denote the number of points in the kth class. The procedure is as follows:

Step 1: **Responses generation**.

Let $y_k = [\underbrace{0,...,0}_{\sum_{i=1}^{j-1} m_i}, \underbrace{1,...,1}_{m_j}, \underbrace{0,...,0}_{\sum_{i=j+1}^{c} m_i}]^T$, $j = 1,...,c$ and $y_0 = [1,1,...,1]^T$ denotes a vector of all ones. Then we can gain orthogonize $\{y_k\}$ by taking y_0 as the first vector to perform Gram-Schmidt process. Then c-1 vectors will be obtained, because y_0 is in the subspace spanned by $\{y_k\}$: $\{\overline{y}_k\}_{k=1}^{c-1}$, $\overline{y}_i^T \overline{y}_0 = 0$, $\overline{y}_i^T \overline{y}_j = 0 \, (i \neq j)$.

Step 2: **Regularized least squares**.

Let x_i still denote a new vector constructed by appending a new element "1" to each original x_i. Then we need to find $c - 1$ vectors $\{\mathbf{a}_k\} \in \mathbb{R}^{n+1}$ by solving the following regularized least squares problem:

$$\mathbf{a}_k = \arg\min_{\mathbf{a}} \left(\sum_{j=1}^{n} (\mathbf{a}^T x_j - y_j^k)^2 + \alpha \|\mathbf{a}\|^2 \right),$$

where $\overline{y}_j^k$ is the jth element of $\overline{y}_k$, $\alpha \geq 0$ is a parameter to control the amounts of shrinkage.

Step 3: **Embedding to $c - 1$ dimensional subspace**.

We can form the transformation matrix A by using the obtained $c - 1$ vectors $\{\mathbf{a}_k\}$ vectors. However, the procedure of transformation ($\mathbf{x} \rightarrow \mathbf{y}$) is slightly different from that of LDA, in which the x must be also appended an element "1": $\mathbf{y} = A^T \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$.

For more details about SRDA, please refer [3].

2.4 Neighborhood Preserving Embedding

Neighborhood Preserving Embedding (NPE) is the linear approximation of Locally Linear Embedding (LLE) and the eigenfunctions of the Laplace Beltrami operator [1]. NPE minimizes the cost function of a local non-linear dimensionality reduction method under the constraint that the mapping from the high-dimensional space to the low-dimensional space is linear.

In NPE, it firstly builds a nearest neighborhood graph G on X by k nearest neighbors or ε neighborhood methods. In this paper, G is obtained by k nearest neighbors method: putting a directed edge from node i to j if x_j is among the k nearest neighbors of x_i. Let W denote the weight matrix of the obtained graph. Then NPE reconstructs the weight W_i the same as the way of LLE. Now we briefly review the reconstruction procedure in [6]. In aforementioned step of calculating the W, we find the k nearest neighbors of one point x_i. However, in many cases, these data points might reside on a nonlinear submanifold. But it might be reasonable for us to assume that the local neighborhood of each point is linear. Therefore, we can characterize the local geometry of these patches by linear coefficients that reconstruct each point from its neighbors [1]. The measure of reconstruction errors can be formally defined as the following cost function [6] and "good" reconstructed weights on the edges can be computed by minimizing this function:

$$\phi(W) = \sum_i \| x_i - \sum_j W_{ij} x_j \|^2$$

with constraints $\sum_j W_{ij} = 1$. The details about how to solve this minimization problem can be found in [6].

Then after the reconstruction, NPE computes the linear projections by solving the following minimization problem:

$$\begin{aligned} &\arg\min_{\mathbf{a}} \mathbf{a}^T XMX^T\mathbf{a} \\ &\text{s.t.} \quad \mathbf{a}^T XX^T\mathbf{a} = 1 \end{aligned} \tag{3}$$

where $M = (I - W)^T (I - W)$, I represents the $n \times n$ identity matrix. For Eq. 3, we can get the transformation matrix A by solving the following generalized eigenvector problem

$$XMX^T\mathbf{a} = \lambda XX^T\mathbf{a} \tag{4}$$

for the top d largest eigenvectors to form A.

We obtain the supervised version of NPE by putting edge to x_i's k neighbors only when these k neighbors are with the same label of x_i.

3 Results

In this section, we investigate the performances of the aforementioned four discriminant analysis methods for classification. All of our experiments have been performed on a P4 1.86GHz Windows XP computer with 2GB memory.

3.1 Experimental Design

In our experiments, we chose 4 publicly available datasets, which are *Leukemia*, *Brain Tumor*, *SRBCT* and *Lung Cancer*.[1] Table 1 summarizes the details about them.

[1] All datasets are available at http://www.gems-system.org/.

Table 1. Statistics of the four datasets

dataset	*size(n)*	*# genes(m)*	*# classes(c)*
Leukemia	72	5327	3
Brain Tumor	90	5920	5
SRBCT	83	2308	4
Lung Cancer	203	12600	5

These datasets are all unbalanced. In fact, microarray datasets, particularly, cancer datasets like these four, are frequently not very balanced. For better comprehension, we now demonstrate the relative frequency of the classes in these datasets:

1. The dataset comprises 47 cases of acute lymphoblastic leukemia (ALL), which contains 38 ALL B-cell and 9 ALL T-cell, and 25 cases of acute myeloid leukemia (AML), whose relative frequency can be simplified as 38/9/25.
2. In the Brain Tumor dataset, there are 60 cases of medulloblastoma, 10 cases of malignant glioma, 10 cases of AT/RT, 4 cases of normal cerebellum, and 6 cases of PNET, whose relative frequency is 60/10/10/4/6.
3. The SRBCT dataset contains 29 cases of Ewing's sarcoma, 11 cases of Burkitt's lymphoma, 18 cases of neuroblastoma and 25 cases of rhabdomyosarcoma, whose relative frequency is 29/11/18/25.
4. In the Lung Cancer dataset, the 203 speciments include 139 samples of lung adenocarcinomas, 21 samples of squamous cell lung carcinomas, 20 samples of pulmonary carcinoids, 6 samples of small-cell lung carcinomas and 17 normal lung samples, whose relative frequency is 139/21/20/6/17.

From above description, we can find these microarray datasets are all unbalanced, which may inevitably distort the impact of the accuracy.

For each dataset, the process of our experiments is designed as follows:

Step 1: For each class, randomly select p (=30%, 40%, 50%, 60%, 70%, 80%) percentage of all samples for training, and the rest are used for testing;

Step 2: Apply the four discriminant analysis methods on the training data to obtain the transformation matrix A. Then both the training and test data are transformed into a d-dimensional subspace by using A;

Step 3: A k nearest neighbor (kNN) is performed on the new dataset, where k is set to be 3 in this paper.

We provide a quantitative evaluation of these discriminant analysis methods via the accuracy of the kNN classifier. A baseline is also provided for comparison, which is obtained by simply performing kNN in the original m-dimensional space. In order to compensate for the random split, each result of our experiments is gained by repeating the aforementioned process 10 times independently.

How we determine the exact d of the subspace in our experiments? For LDA and SRDA, we simply transform the original data into a (c-1)-dimensional subspace, which is already very small. For both S-NPE and LSDA, we firstly transform the original data into a large enough subspace, i.e. 50 percent of the original dimensionality. Then remove the eigenvectors with small or zero eigen-values. In our experiments, we set that threshold eigen-value to be 10^{-6}. In our experiments, we found many

obtained hybrid-genes through dimensionality reduction methods with so small eigenvalues, and the dimensionality of the subspace are usually smaller than 50.

We select 3 nearest neighbors to build the graph in LSDA and S-NPE. By referring to [2,3], the α of LSDA and SRDA are set to be 0.05 and 0.1 respectively.

Then why we select 3 nearest neighbors instead of other numbers of neighbors? In fact, it is still unclear how to define the locality theoretically [1]. That is, it is still unclear how to select the parameter k in a principled manner. From our observations and experience, setting k to be 3 is a good compromise plan to gain promising performances of algorithms such as NPE and LSDA on all of the four microarray datasets.

Table 2. Accuracy (%) (mean reduced dimensionality) and computational time (s) on the four datasets

	Dataset	LDA	LSDA	S-NPE	SRDA	Baseline
Accuracy	Leukemia	87.36(2)	68.68(28)	80.60(13)	**88.43(2)**	78.55
	Brain Tumor	89.01(4)	77.74(35)	87.17(16)	**90.14(4)**	80.84
	SRBCT	97.23(3)	77.46(33)	79.82(17)	**97.95(3)**	76.81
	Lung Cancer	93.24(4)	83.70(63)	90.50(31)	**94.84(4)**	89.33
Times (s)	Leukemia	0.1013	0.2125	0.2300	**0.0826**	0.4401
	Brain Tumor	0.1813	0.3310	0.3490	**0.1378**	0.7818
	SRBCT	0.0609	0.1232	0.1203	**0.0513**	0.2206
	Lung Cancer	0.9620	1.8716	1.9638	**0.6656**	8.0469

3.2 Results and Discussion

Table 2 summarizes the results of our experiments on the four datasets. Limited to the length, we average the accuracy of the six different p percentages of splitting the dataset. From Table 2, we find that the performances of SRDA are always the best of the four methods and much better than those of baseline, from both the accuracy and computational time. The performance of LSDA is the worst, which shows LSDA seems not suitable to take the task of dimension reduction for microarray data. Though the accuracy of LDA and S-NPE is not that high as that of SRDA, they sharply reduce the time-cost of the classifier, which is also very valuable. We visualize the change of the accuracy and computational time under different values of p in Figs. 1-2. There are some issues thrown out by these results, and we will discuss them respectively and carefully below.

Why SRDA and LDA are the best ones? Our experimental results show SRDA and LDA perform best, and SRDA is more excellent than LDA. The reasons may be:

1) From the biological view, many diseases are usually determined by a few special genes. That is, the hybrid genes extracted by SRDA capture enough information for classification, which may be a valuable virtue of SRDA. That also means more valuable referenced information can be gained by using SRDA with less time cost.

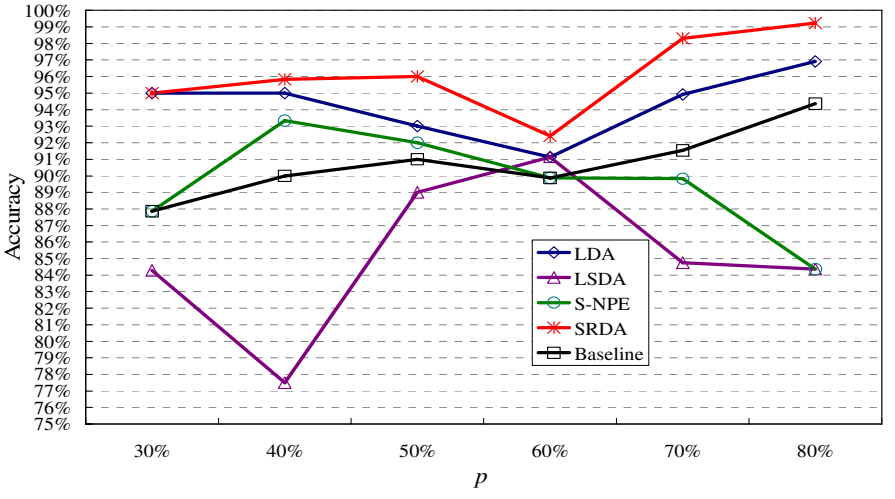


Fig. 1. Accuracy curve on Lung Cancer dataset

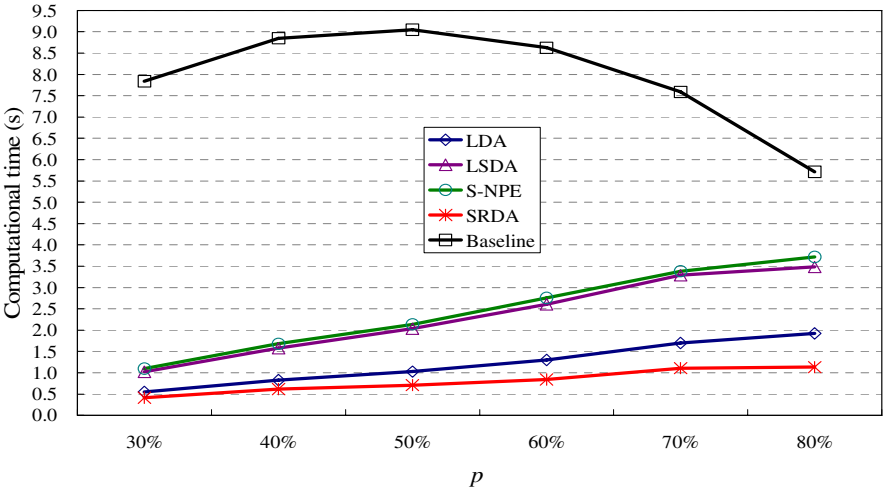


Fig. 2. Computational time curves on Lung Cancer dataset

2) From the view of machine learning community, the aim of dimensionality reduction is to embed high-dimensional data samples in a low-dimensional space while most of 'intrinsic information' contained in the data is preserved. That means, low-dimensional presentation of original data may provide enough information for the aim of classification.

Why the performance of LSDA is so poor? The possible reason is the noise from those unrelated genes. From above discussion, there are always only a few genes are related to a special disease. In the procedure of LDSA, two graphs: *within-class graph* G_w and *between-class graph* G_b, are built through k nearest neighbors method. These neighbors are determined by using euclidean distance under the original high-dimensional space in our experiments. That means, the noise from those unrelated genes may result in the 'poor' quality of neighbors, i.e., these neighbors are not that near if omitting those unrelated genes.

Why the performance of LSDA is so fair? Since, in this paper, the information from labels are taken use of by putting larger weight to the neighbors with the same label of x_i than those in different class(es), the valuable information may not be taken full use of. Another possible reason is the curse from these poor qualities of neighbors, the same with that of LDSA.

4 Conclusion

In this paper, we firstly introduced three novel supervised dimension reduction techniques, Locality Sensitive Discriminant Analysis (LSDA), Spectral Regression Discriminant Analysis (SRDA), and Supervised Neighborhood Preserving Embedding (S-NPE), for microarray data classification. Results of our experiments showed that SRDA is an efficient and effective discriminating technique for microarray data analysis. We also analyzed these results detailedly, and gave our reasons for some important problems uncovered by these results. In future, we will explore the discriminating power of additional discriminant analysis methods.

Acknowledgements. The research work described in this paper was supported by grants from the National Natural Science Foundation of China (Project No. 10601064).

References

1. He, X., Cai, D., Yan, S., Zhang, H.J.: Neighborhood Preserving Embedding. In: Proc. of 10th IEEE Int. Conf. on Computer Vision, Beijing, China (2005)
2. Cai, D., He, X., Zhou, K., Han, J., Bao, H.: Locality Sensitive Discriminant Analysis. In: Proc. of 2007 Int. Joint Conf. on Artificial Intelligence, Hyderabad, India (2007)
3. Cai, D., He, X., Han, J.: SRDA: An Efficient Algorithm for Large Scale Discriminant Analysis. IEEE Transactions on Knowledge and Data Engineering 20, 1–12 (2008)
4. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Eugenics 7, 179–188 (1936)
5. Van der Maaten, L.J.P.: An Introduction to Dimensionality Reduction Using Matlab. Report MICC 07-07. Maastricht University, Maastricht, the Netherlands (2007)
6. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by Locally Linear Embedding. Science 290, 2323–2326 (2000)
7. Chee, M., Yang, R., Hubbell, E., et al.: Accessing genetic information with high-density DNA arrays. Science 274, 610–614 (1996)

8. Schena, M., Shalon, D., Davis, R.W., et al.: Quantitative monitoring of gene expression patterns with a complementary DNA micro-array. Science 270, 467–470 (1995)
9. Raychaudhuri, S., Sutphinb, P.D., Changa, J.T., et al.: Basic microarray analysis: grouping and feature reduction. TRENDS in Biotechnology 19, 189–193 (2001)
10. Bowman, C., Aruliah, D., Fan, G., et al.: Nonlinear Dimension Reduction for Microarray Data (Small n and Large p). In: Proc. of the 1st Fields–MITACS Industrial Problems Workshop, Toronto, pp. 1–15 (2006)
11. Alter, O., Brown, P.O., Botstein, D.: Singular value decomposition for genome-wide expression data processing and modeling. Proc. of the National Academy of Sciences 97, 10101–10106 (2000)
12. Dai, J.J., Lieu, L., Rocke, D.: Dimension Reduction for Classification with Gene Expression Microarray Data. Statistical Applications in Genetics and Mol. Biol. 5, 1–15 (2006)
13. Shi, C., Chen, L.: Feature Dimension Reduction for Microarray Data Analysis Using Locally Linear Embedding. In: Proc. of APBC 2005, pp. 211–217 (2005)
14. Dawson, K., Rodriguez, R.L., Malyj, W.: Sample phenotype clusters in high-density oligonucleotide microarray data sets are revealed using Isomap, a nonlinear algorithm. BMC Bioinformatics 6 (2005)
15. Nilsson, J., Fioretos, T., Höglund, M., et al.: Approximate geodesic distances reveal biologically relevant structures in microarray data. Bioinformatics 20, 874–880 (2004)
16. Ye, J., Li, T., Tao, X., Janardan, R.: Using Uncorrelated Discriminant Analysis for Tissue Classification with Gene Expression Data. IEEE/ACM Trans. Comput. Biology Bioinform. 1, 181–190 (2004)
17. Cai, D., He, X., Han, J.: Training Linear Discriminant Analysis in Linear Time. In: Proc. 2008 Int. Conf. on Data Engineering (ICDE 2008), Cancun, Mexico (2008)

Propositionalisation of Profile Hidden Markov Models for Biological Sequence Analysis

Stefan Mutter, Bernhard Pfahringer, and Geoffrey Holmes

Department of Computer Science
The University of Waikato
Hamilton, New Zealand
{mutter,bernhard,geoff}@cs.waikato.ac.nz

Abstract. Hidden Markov Models are a widely used generative model for analysing sequence data. A variant, Profile Hidden Markov Models are a special case used in Bioinformatics to represent, for example, protein families. In this paper we introduce a simple propositionalisation method for Profile Hidden Markov Models. The method allows the use of PHMMs discriminatively in a classification task. Previously, kernel approaches have been proposed to generate a discriminative description for an HMM, but require the explicit definition of a similarity measure for HMMs. Propositionalisation does not need such a measure and allows the use of any propositional learner including kernel-based approaches. We show empirically that using propositionalisation leads to higher accuracies in comparison with PHMMs on benchmark datasets.

1 Introduction

Traditionally, research in machine learning has focussed on fixed length attribute-value representations of data. Hence classifiers for this kind of data have undergone an intense optimisation process. In many areas, however, it is easier to represent data as structured terms. Examples include areas such as relational data mining or bioinformatics. In the field of classification from structured data [1], there are three common ways of dealing with this kind of representation: relational rule learning [2], kernel approaches [3,4] or propositionalisation [5,6,7].

In this paper we use propositionalisation to generate attribute-value representations of varying complexity for Multiple Sequence Alignment. The Multiple Sequence Alignment is represented using a Profile Hidden Markov Model (PHMM) [8,9]. A PHMM is a generative, probabilistic, graphical model. Using propositionalisation allows the use of complex PHMMs in a simple, discriminative way. Kernel methods have also been successfully applied to obtain a discriminative description of Hidden Markov Models (HMMs) [3,10]. However, they require the definition of an explicit similarity measure between different HMMs. Propositionalisation approaches do not need such a pre-defined similarity measure.

The remainder of this paper is organised in the following way. Section 2 discusses Hidden Markov Models in general and their use in sequence classification.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 278–288, 2008.

Section 3 describes kernel methods for structured classification emphasising kernel methods for biological sequence data, whereas Section 4 explains propositionalisation approaches. In Section 5 we present our experimental setup and results. Conclusions and future work are presented in the final section.

2 Hidden Markov Models

Hidden Markov Models (HMM) are widely used for sequence analysis, especially in natural language processing and Bioinformatics [11,9,12]. In Bioinformatics HMMs have been successfully applied to gene-finding, phylogenetic analysis and protein secondary structure prediction. An HMM represents, in general, a probability distribution over sequences. It is a generative, probabilistic, graphical model.

Profile HMMs (PHMMs) are a special kind of HMM for representing Multiple Sequence Alignments. They allow an alignment to be trained from unaligned sequences. Figure 1 shows three unaligned amino acid sequences which can be used as input for a PHMM. This is exactly the kind of input data we use in our experiments.

```
MMFFADDAAAE
MMFFARRNSSTNNRREDPFMLWE
MMFFAE
```

Fig. 1. Three short, unaligned (artificially created) amino acid sequences

A Multiple Sequence Alignment consists of three or more either DNA, RNA or amino acid sequences. In this paper we align amino acid sequences globally. PHMMs allow the construction and representation of a Multiple Sequence Alignment and offer the advantage of being a probabilistic model. They were introduced by Krogh et al. [8] especially for protein modelling, and are widely used in bioinformatics [9] to represent protein families. Their underlying graphical structure suits the characteristics of an alignment. The structure for our PHMM closely follows the popular PHMM model of the HMMER software [13], and differs slightly from the original proposition of Krogh et al [8]. There are no transitions from an insert to a delete state and vice versa as these transitions are very unlikely in the biological domain. Each position in a PHMM consists of a match, insert and delete state, exceptions are the first position and the last position. In the first position there is only a match and an insert state, and in the last position there is only a match state. Figure 2 shows a PHMM with four match states.

As stated previously, a PHMM can represent a Multiple Sequence Alignment. Figure 3 shows such a Multiple Sequence Alignment for the unaligned sequences from Figure 1. Following convention in the biological literature, deletions are marked with a '-'. In addition matching parts of the sequence are indicated by

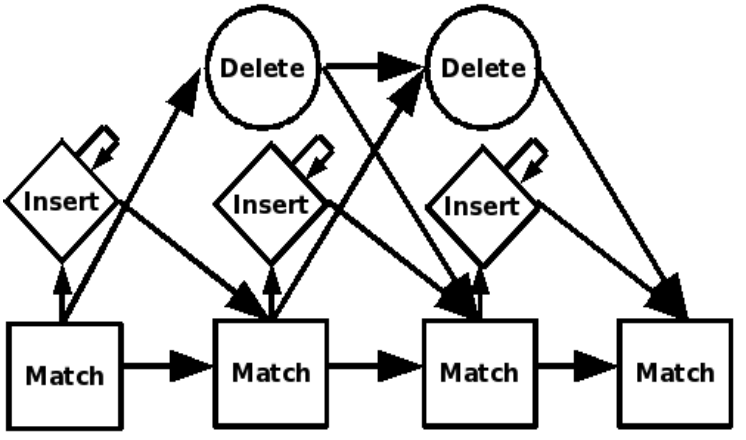


Fig. 2. The general structure of a Profile Hidden Markov Model (PHMM) of length 4

```
MMFFA           DDAAA--E
MMFFArrnsstnnrrEDPFMLWE
MMFFA           -------E
```

Fig. 3. A global Multiple Sequence Alignment for our example sequences. This is one part of the output of a trained PHMM.

upper case letters, whereas insertions are displayed with lower case letters. The graphical structure of a PHMM easily allows this representation and therefore to identify conserved regions, so that they can be displayed below one another. A Multiple Sequence Alignment, like the one presented in Figure 3, is part of a trained PHMM's output. However, because it is a probabilistic model, its representation of an alignment is much richer. We will make use of the alignment itself, as well as of the probabilistic features of a PHMM.

In this paper, PHMMs are trained from unaligned sequences using the Baum-Welch algorithm [9]. These trained models can be used directly for classification by calculating of the probability of a test sequence given the model. We use the performance of these models as a baseline comparison.

During training, a PHMM for each class is constructed separately. Therefore each PHMM is only trained on its positive instances. As Jaakkola et al. [3] point out, discriminative tasks such as classification might benefit from using the negative examples as well. They extract a kernel description of a HMM and use this for classification by a support vector machine. We achieve discriminative modelling by extracting features from the PHMM through propositionalisation and subsequently training a propositional learner. This approach is not restricted to kernel-based classifiers.

3 Kernel Approaches

Kernel approaches are popular and extensively used in the machine learning community. Using kernel methods on structured data [4,3,10] has led to accurate classifiers [1]. Nevertheless, defining an appropriate kernel function is still a difficult problem [1].

For biological sequence analysis there are three different kernel approaches that are of special interest: general graph kernels used on a problem-specific graph for proteins [4], kernels defined over an arbitrary HMM [10] and Fisher kernels defined on an HMM representing a protein family [3].

Borgwardt et al. [4] define a random walk graph kernel over a specially defined graphical model for proteins. Each graph represents one protein and similarity is calculated by a graph kernel. Our approach differs in that respect, because we use a general model, a PHMM, to model the alignment of proteins. We align different proteins using one graphical model. In the PHMM the similarity is expressed by a probability. We propositionalise our structured model and therefore do not need to define a similarity measure.

Recently a probabilistic pair-wise kernel over HMMs has been used for spectral clustering of time series data [10]. This probability product kernel measures the distance between two HMMs each representing a time series. The clustering of the time series data is then based on this similarity measure. This semi-parametric approach allows the representation of the inherent structure of the time series data without being excessively strict with assumptions about the overall distribution. In the same way, we assume an HMM structure for all instances belonging to the same class, but make no assumption about the overall distribution. In our approach a PHMM is trained separately for each class.

Jaakkola et al. [3] define the Fisher score over a trained HMM for a protein family and use it as a kernel function in their support vector machine approach. They successfully use a trained generative model as an intermediate step to obtain a discriminative model by explicitly defining a similarity function. The HMMs used by Jaakkola et al. [3] are PHMMs. However, they allow transitions from insert to delete states and vice versa and a sequence alignment can begin and end with a match, insert or delete. In addition they align sequences locally, whereas we align sequences globally. Their PHMM is described by Karplus et al. [14]. In our approach we use the generative model directly to extract features for discriminative modelling. In addition, their approach is restricted to kernel based classifiers whereas we can use the extracted features as input for any propositional learner.

A drawback of all kernel approaches is the need to explicitly define a similarity measure with the kernel function. Propositionalisation does not have this requirement.

4 Propositionalisation

Propositionalisation transforms complex, structured representations of data into a fixed length representation involving attribute-value pairs [6]. Therefore, it

de-couples model from feature construction and introduces a new degree of flexibility and a wide range of possible features to be constructed from the more complex representation. Traditionally machine learning has focused on propositional learners, thus they are biased to this representation.

In this paper, propositionalisation means to take a probabilistic, graphical model and transform it into a fixed length attribute-value representation. To state it clearly, the suggested technique uses PHMMs as input and creates a fixed number of attributes and corresponding values from them. The major contribution of this paper is the examination of different ways to propositionalise a PHMM. The complexity of the different propositionalisations is varying. In addition, most of the propositionalisations can be used with HMMs in general as well.

By de-coupling model from feature construction we do not need to explicitly define a similarity measure for graphs like in kernel based methods. The representation resulting from propositionalisation can be used as input for a wide range of classifiers including but not limited to kernel-based classifiers. We propositionalise PHMMs and show extensions to propositionalise general HMMs. This propositionalisation and the subsequent propositional classification step require only a small overhead compared to the training of the underlying PHMM. As training time of the PHMM is a dominant factor, it is favourable to reduce it. Training a PHMM with smaller sequences and subsequently applying propositionalisation can achieve this. Using this approach the propositional learner might be able to cope with less information. We will investigate this potential later.

The key idea behind our propositionalisation approach is simply to take advantage of the canonical form of the underlying graph of a PHMM. The aim of the propositionalisation is to get a fixed length attribute-value representation of the datatset. This new representation should be as simple as possible in order not to include too many, potentially irrelevant features. On the other hand, a representation which is too simple might not capture all the information that is essential for an accurate classification. Therefore, we use propositionalisations of different complexities and compare the results.

In order to get a propositional representation for an input sequence, we calculate the most probable path for the sequence through the trained PHMM. This path is also known as the Viterbi path. On the Viterbi path each match state is visited except when it is skipped by a delete state. In the case of a match, it can be followed by a sequence of insertions. This directly tells us how to get a propositional representation of the Viterbi path by exploiting the fact that a PHMM allows the representation of the Viterbi path of any sequence in a fixed length way. The process proceeds as follows:

1. Create a nominal attribute for each match state. The values of the nominal attributes are the emitted symbols plus an extra symbol representing a deletion. In our PHMM the emission alphabet comprises the symbols defining amino acids.

2. Create a numerical attribute for each insert state. Set the value of the attribute according to the number of times this state is visited in the Viterbi path, i.e. the number of insertions at that point.

Thus every unit of a match, an insert and a delete state is converted into two attributes. Figure 4 shows the propositional representation for the simple example Multiple Sequence Alignment from Figure 3.

```
M,0,M,0,F,0,F,0,A, 0,D,0,D,0,A,0,A,0,A,0,-,0,-,0,E
M,0,M,0,F,0,F,0,A,10,E,0,D,0,P,0,F,0,M,0,L,0,W,0,E
M,0,M,0,F,0,F,0,A, 0,-,0,-,0,-,0,-,0,-,0,-,0,-,0,E
```

Fig. 4. The propositional representation of the example Multiple Sequence Alignment

This representation is highly motivated by the graphical form of a PHMM and is simple. It can be used by any propositional learner that can handle nominal and numeric attributes. Hence we use a generative model for feature generation and convert it into a discriminative one via propositionalisation for the classification task.

The classification problems considered in this research are multi-class classification problems. For each class we train one PHMM using only the sequences belonging to this class. To generate a propositional representation of a sequence, we calculate the Viterbi paths through the trained PHMMs of all classes and combine the resulting propositional representations from each PHMM into one representation.

Using only the Viterbi path to propositionalise leads to a simple representation that is limited to PHMMs. In the remainder of this paper we will refer to this simple form of propositional representation as mode 1. It is preferable to have a simple way to propositionalise not only PHMMs, but HMMs in general. In order to achieve this goal, we have to make sure to generate fixed length descriptions independent of the length of the path. The following modes can all be applied to HMMs in general and are not limited to PHMMs. Thus, as long as we do not include the simple mode 1 into the propositional representation, our approach can be applied to HMMs in general. Mode 2 converts the whole HMM into a single numerical attribute. It represents the score of the Viterbi path for each sequence given the HMM. In the same way, we construct another numeric attribute in mode 3 representing the score of the sequence given the HMM. This is the sum of the scores of all possible paths, not only the score of the Viterbi path. These are two very simple ways to propositionalise general HMMs. However, there is much more information present in the underlying HMM. In mode 4 we create a numeric attribute for each state of the HMM representing the score of the state given the sequence. This representation has a fixed length for all input sequences. In the same way, the probability of each state given the sequence is used in mode 5. An advantage of a propositional approach is that we are now able to put together all different modes in all possible combinations.

Table 1. The propositional algorithms used in the experiments

Propositional Algorithm	Parameters
Naive Bayes	
k-Nearest Neighbours	uses best k between 1 and 100
Random Forest	with 100 trees and various numbers of features
SMO	with linear, polynomial and RBF kernel
Bagging	using 100 iterations with an unpruned J48
AdaBoost	using 100 iterations with a pruned J48

5 Experiments

For our experiments we set up the PHMM in the following way. In general, and in the absence of prior knowledge, the PHMM has as many match states as the average number of residues in the training sequences as suggested by Durbin et al. [9]. If the sequence length is artificially restricted for a training set, the number of match states equals the cutoff number for the sequence length. The emission probabilities in the insert and match states are initialised uniformly. We only train emissions in match states. All transitions are initialised uniformly as well. This is a very simple initialisation, because we do not assume any prior knowledge. Our purpose is to demonstrate the extent to which classification can be improved by adding a subsequent propositionalisation step over classification purely based on the generative PHMM.

The PHMM is trained from unaligned sequences using the Baum-Welch algorithm, a special case of the EM algorithm. It guarantees convergence to a (local) optimum. The convergence criterion is a sufficiently small change in the log-odds score relative to a random model. This score is normalized by the number of residues in a sequence[1]. Subsequently the trained PHMM is used for propositionalisation. In each classification problem, we train one PHMM per class label.

We test our propositional representations on various propositional learners. See Table 1 for an overview.

The datasets are taken from the Protein Classification Benchmark Collection [15]. They belong to the *3PGK_Protein_Kingdom_Phylum* set of sequences. It consists of 10 binary classification problems which map protein sequences into kingdoms of life based on phyla. The database provides training and test sets and reports benchmark results for the Area Under the ROC Curve (AUC) for these problems. We chose the two classification problems Eukaryota/Euglenozoa and Bacteria/Proteobacteria for our investigations. In the remainder of the paper we will refer to the former as 3pgk6. The latter will be called 3pgk4. The reported benchmark results in the data collection suggest that these two binary classification tasks are the most challenging ones in the *3PGK_Protein_Kingdom_Phylum* problem domain. For all datasets the best and worst AUC values are reported. Considering only the best AUC values for all ten classification problems in the

[1] The threshold used was proposed by A. Krogh in a personal e-mail communication.

3PGK_Protein_Kingdom_Phylum data, 3pgk4 best AUC result is 0.93 and the one for 3pgk6 is 0.92. No other dataset has a lower AUC value for its best performing classification model. In addition, all experimental results for 3pgk4 and 3pgk6 achieve at least an AUC value of 0.56. There is only one dataset with a lower overall worst AUC value.

The training set of 3pgk6 consists of 82 training instances whereas the test set has 49 instances. The average sequence length in the training set is 414 and 406 in the test set. The training set and the test set each have 44 positive instances. In the 3pgk4 training set there are 70 instances, 27 of them are positive. The test set consists of 61 instances. Of these, 31 instances belong to the positive class. In the training set the average sequence length is 413, whereas it is 408 in the test set. For both datasets sequences vary in general between 358 and 505 residues.

These datasets are used twice in the experiments. In a first setting we use the full sequence information, whereas in the second setting we artificially reduce the length of the sequence to the first 100 residues. All experiments are performed using the WEKA machine learning workbench [16].

We also present the result from a pure PHMM approach where the PHMM is used directly for classification. In this setting a PHMM for each class label is built as well. The log-odds score of the test sequence given the model is used as a basis for classification.

In all results we only show the propositional learner and its respective parameters which performed best. First, we present the results for the 3pgk6 dataset. In the first experiment we use the full sequence information. The results are shown in Table 2.

Table 2. The accuracies for the 3pgk6 dataset. The sequence length is not restricted. The propositional learner is a Support Vector Machine with RBF Kernel, γ is set to 1. We fit logistic models to the output. The baseline accuracy of the PHMM without propositionalisation is included as well.

modes' combinations	SVM accuracy	PHMM baseline accuracy
{1}, {1,2}, {1,3},{1,2,3},{4},{5}, {1,4},{1,5},{1,2,3,4},{1,2,3,5}	89.80%	81.63%

All propositional representations outperform the pure PHMM approach. In all settings a Support Vector Machine with an RBF Kernel works best. All representations, simple or complex achieve the same accuracy.

In a second experiment we restricted the sequence length artificially to the first 100 residues. We use the same combinations of modes as in the previous experiment. Table 3 summarises the results. In this case as well, the different combinations of modes lead to the same accuracy result. The best performing propositional learner is again a Support Vector Machine with an RBF Kernel. Using propositionalisation in a setting were information is restricted does not lead to a decrease in accuracy, whereas for the pure PHMM approach it does.

Table 3. The accuracies for the 3pgk6 dataset. The sequence length is restricted to the first 100 residues. The propositional learner is a Support Vector Machine with RBF Kernel, γ is set to 1. We fit logistic models to the output. When mode 5 is used on its own, γ is set to 100.

classifier	accuracy
PHMM	79.59%
all propositional learners	89.80%

Table 4. The accuracies for the 3pgk4 dataset. The sequence length is not restricted. The baseline accuracy of the PHMM is 68.85%.

combination of modes	propositional learner	accuracy
1	**Random Forest with 100 trees, 300 features**	**93.44%**
1,2	Random Forest with 100 trees, 200 features	90.16%
1,3	Random Forest with 100 trees, 200 features	90.16%
1,2,3	**Random Forest with 100 trees, 200 features**	**93.44%**
4	k-nearest neighbours	86.69%
5	Random Forest with 100 trees, 4 features	77.05%
1,4	k-nearest neighbours	86.69%
1,5	Random Forest with 100 trees, 15 features	73.77%
1,2,3,4	k-nearest neighbours	86.69%
1,2,3,5	Random Forest with 100 trees	77.05%

Table 4 gives on overview of the results for the 3pgk4 dataset when the full sequence information is used. Again all propositional representations outperform the pure PHMM based approach. For this dataset the simpler propositionalisations perform considerably better than the baseline classifier. In addition, they outperform more complex propositional representations. Best accuracy is achieved when we use the Viterbi path for propositionalisation or when we use the Viterbi path and include its score and the score of the sequence given the PHMM. However, when using the path information only, the Random Forest needs more features to achieve the same accuracy.

In the opposite case, when information is limited, the results shown in Table 5 indicate that additional information is helpful. The pure PHMM approach does not suffer from a lack of information. It achieves the same accuracy as before when the full sequence was used. The propositional approaches still outperform solely PHMM based classification. However, their accuracies are worse. In this setting the more complex propositionalisations outperform the simpler ones.

However there are cases, when information is limited, where the pure PHMM classification results in a slightly better accuracy than the classifiers using propositionalised data as the following experiment on the 3pgk6 dataset shows. In this case, we restrict the sequence length to the first 150 residues. The baseline accuracy of the PHMM is 91.84%. The most accurate propositional learner is a Support Vector Machine with RBF Kernel, γ is set to 1. Additionally, it fits

Table 5. The accuracies for the 3pgk4 dataset. The sequence length is restricted to 100 residues. The baseline accuracy of the PHMM is 68.85%.

combination of modes	propositional learner	accuracy
1	Bagging	80.33%
1,2	Random Forest with 100 trees, 15 features	80.33%
1,3	k-nearest neighbours	78.69%
1,2,3	k-nearest neighbours	78.69%
4	k-nearest neighbours	81.97%
5	Random Forest with 100 trees, 4 features	81.97%
1,4	**Random Forest with 100 trees, 15 features**	**83.61%**
1,5	**linear SMO, complexity 0.001**	**83.61%**
1,2,3,4	**Random Forest with 100 trees**	**83.61%**
1,2,3,5	**linear SMO, complexity 0.001**	**83.61%**

logistic models to the output. We used 4 different experimental setups. First, we propositionalise the dataset according to mode 1. In the second experiment, a combination of modes 1 and 2 is used for propositionalisation, whereas the third experiment employs modes 1 and 3. The last experiment is conducted with modes 1, 2 and 3. All these experiments lead to the same resulting accuracy of 89.80%. The pure PHMM approach outperforms the propositionalisation in this case. However, compared to the previous results, the difference in accuracies between the baseline and the propositional learners is not as large.

6 Conclusions and Future Work

In this paper we introduced a simple way to propositionalise a PHMM and extended it to ways of propositionalising general HMMs. We compared different propositional representations with each other and the classification performance of the PHMM as a baseline.

We showed that a simple propositionalisation of a complex, generative PHMM leads, most of the time, to better results on two benchmark datasets than a purely PHMM based classification. In cases where information is limited, more complex propositional representations perform better or equally as well as their simpler counterparts. These results indicate that propositionalisation of a PHMM is able to outperform a solely PHMM based classification approach. However, the findings on these benchmark datasets need to be verified on other datasets as well. Consequently, as a next step, we will test our approach on more and larger datasets.

In addition, we will have a closer look at the idea of reducing training time for a PHMM without losing overall discriminative power. Instead of limiting the number of residues to train smaller PHMMs, we anticipate training a PHMM ahead of full convergence of the Baum-Welch algorithm, and then using propositionalisation. This leads to a considerable speed-up of the training process. Propositionalisation might be able to compensate for a decrease in accuracy.

References

1. Karunaratne, T., Boström, H.: Learning to classify structured data by graph propositionalization. In: Proceedings of the 2nd IASTED International Conference on Computational Intelligence, pp. 283–288. IASTED/ACTA Press (2006)
2. Zaki, M.J., Aggarwal, C.C.: Xrules: An effective structural classifier for xml data. In: KDD 2003: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 316–325. ACM, New York (2003)
3. Jaakkola, T., Diekhans, M., Haussler, D.: Using the fisher kernel method to detect remote protein homologies. In: Proceedings of the 7h International Conference on Intelligent Systems for Molecular Biology, pp. 149–158. AAAI Press, Menlo Park (1999)
4. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S.V.N., Smola, A.J., Kriegel, H.: Protein function prediction via graph kernels. Bioinformatics 21(1), 47–56 (2005)
5. Pfahringer, B., Holmes, G.: Propositionalization through stochastic discrimination. In: Proceedings of the Work-in-Progress Track at the 13th International Conference on Inductive Logic Programming, Department of Informatics, University of Szeged, Hungary, pp. 60–68 (2003)
6. Krogel, M.-A., Rawles, S., Železný, F., Flach, P.A., Lavrač, N., Wrobel, S.: Comparative Evaluation of Approaches to Propositionalization. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS, vol. 2835, pp. 197–214. Springer, Heidelberg (2003)
7. Lachiche, N.: Good and Bad Practices in Propositionalisation. In: Bandini, S., Manzoni, S. (eds.) AI*IA 2005. LNCS, vol. 3673, pp. 50–61. Springer, Heidelberg (2005)
8. Krogh, A., Brown, M., Mian, I., Sjölander, K., Haussler, D.: Hidden markov models in computational biology: Applications to protein modelling. Journal of Molecular Biology 235(5), 1501–1531 (1994)
9. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge (1998)
10. Jebara, T., Song, Y., Thadani, K.: Spectral clustering and embedding with hidden markov models. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 164–175. Springer, Heidelberg (2007)
11. Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
12. Bystroff, C., Thorsson, V., Baker, D.: Hmmstr: A hidden markov model for local sequence-structure correlations in proteins. Journal of Molecular Biology 301, 173–190 (2000)
13. Eddy, S.: Hmmer user's guide: biological sequence analysis using profile hidden markov models (1998), `http://hmmer.wustl.edu/`
14. Karplus, K., Barrett, C., Hughey, R.: Hidden markov models for detecting remote protein homologies. Bioinformatics 14(10), 846–856 (1998)
15. Sonego, P., Pacurar, M., Dhir, S., Kertész-Farkas, A., Kocsor, A., Gáspári, Z., Leunissen, J.A.M., Pongor, S.: A protein classification benchmark collection for machine learning. Nucleic Acids Research 35(Database-Issue), 232–236 (2007)
16. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

Improving Promoter Prediction Using Multiple Instance Learning

P.J. Uren, R.M. Cameron-Jones, and A.H.J. Sale

School of Computing and Information Systems
Faculty of Science, Engineering and Technology
University of Tasmania, Hobart and Launceston, Tasmania, Australia
{Philip.Uren,Michael.CameronJones,Arthur.Sale}@utas.edu.au

Abstract. Promoter prediction is a well known, but challenging problem in the field of computational biology. Eukaryotic promoter prediction, an important step in the elucidation of transcriptional control networks and gene finding, is frustrated by the complex nature of promoters themselves. Within this paper we explore a representational scheme that describes promoters based on a variable number of salient binding sites within them. The multiple instance learning paradigm is used to allow these variable length instances to be reasoned about in a supervised learning context. We demonstrate that the procedure performs reasonably on its own, and allows for a significant increase in predictive accuracy when combined with physico-chemical promoter prediction.

1 Introduction and Biological Context

Deoxyribonucelic acid (DNA) stores the instructions for building complex biological organism. Simply speaking, this information is arranged in genes, many of which (but not all) code for proteins — the general functional units of biological systems. DNA is a long polymeric molecule constructed from monomers (of which there are four). Within computational fields, this is commonly represented as a string of letters from the alphabet of A, C, T and G (representing the bases Adenine, Guanine, Cytosine and Thymine); the length unit is base pairs (or *bp*) and is analogous to simple string length. Although DNA is a double helix *in vivo*, it is common to give the sequence of only one strand as the bases bind complementarily; one strand can be inferred from the other. For the purposes of this paper, we will concentrate on the eukaryotic *homo sapiens* genome. Relatively little of the human genome is actually accounted for by genes [1], making the localisation of them highly important. Furthermore, the expression (activity) level of genes is governed by complex regulatory networks involving the binding of proteins called transcription factors to the DNA molecule. This regulatory network ensures the correct temporal, spatial and contextual expression of each gene. A promoter can be thought of as a gene header — a short portion of DNA which is not transcribed, but allows the cellular transcriptional machinery to recognise and bind at the correct location on the DNA molecule. The point within the promoter after which the sequence is transcribed is called the transcription start site (or *TSS*).

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 289–299, 2008.

Transcription factor binding sites are often clustered around, or within, the promoter region [2-5]. Transcription factors (regulatory proteins) bind to these sites, modifying the expression level of the gene. A common method for identifying transcription factor binding sites is via position weight matrices (PWMs). Position weight matrices model the likelihood of a given nucleotide appearing at a given location within a binding site; one matrix is needed for each transcription factor. Given a putative binding site, the model gives the quality of match and a threshold can be used to predict sites. This paper explores a promoter representational scheme based around the location of these transcription factor binding sites.

Prestridge [3] describes an approach based on representing promoters by the location of transcriptional elements. This approach uses a collection of position weight matrices to build a profile of the promoter region. For each transcriptional element considered, the ratio density of occurrences within promoter sequences, as compared to non-promoter sequences, is calculated. A score for a putative promoter is calculated by summing the density scores from the profile which match the transcriptional elements found within the sequence in question. In contrast to the method proposed here however, this does not take into account the relative positioning of sites.

Kondrakhin and colleagues [2] propose a similar technique to that of Prestridge, although using consensus sequences rather than position weight matrices to identify binding sites. The salient difference is that they consider the localisation of binding sites. They split the promoter into regions and construct a two-dimensional matrix representing the occurrence of each binding site within each region. Classification is performed from this matrix via a weighted sum using a threshold.

Other work has also investigated the density of particular oligonucleotides in promoter regions [6, 7]. Within the work of Narang et al. [7] a statistical model is created from a dataset of positive instances without the need for a collection of weight matrices (or other models of the motifs sought). Negative examples or a background genomic model are also not required. The fact that this method does not require pre-existing models of the motifs sought is a major advantage, as many transcription factors either do not have (or have poorly supported) models.

The efficacy of characterising promoters by the distribution of salient motifs within their primary sequence has clearly been established in previous work. The focus here is on a formulation which is amenable to solution within the classic supervised machine learning framework. To this end the next section explores a possible approach utilising the common attribute-vector representation.

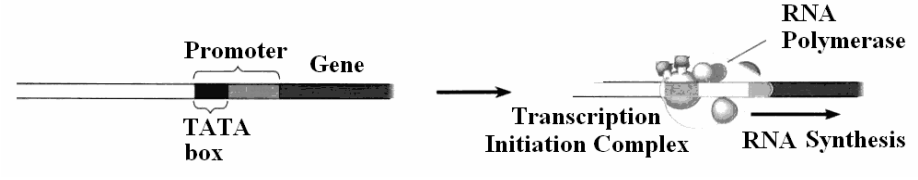


Fig. 1. The process of transcription showing the binding of transcription factors to the DNA molecule within the promoter region (adapted from [8])

2 Promoter Prediction Using Transcription Factor Binding Sites

A significant number of approaches have previously been presented aimed at solving the promoter prediction problem [9-17]. Bajic [15] et al. report that none of the programs they tested in their review were able to produce a combined sensitivity and positive predictive value of more than 65%, with many falling well below this. We use the same accuracy measures here as used in [15] and [18]. Here we explore a new approach to promoter prediction utilising the existing idea of describing a promoter by the location and name of transcription factor binding sites which fall within it. A relatively simple mechanism is employed for locating these binding sites; the putative promoter region is scanned using position weight matrices (PWMs). This leads to a new consideration: the problem of variable length instances. If one scans the promoter sequence using a weight matrix this produces a series of matches (i.e. there may be zero or more sites that exceed the threshold). Repeating the process searching for binding sites for other distinct transcription factors produces a heterogeneous set of matches. This set is the collection of attributes used to describe a single promoter, or instance, within the dataset. Although it is conceivable that the number of distinct factors for which binding sites are sought is known *a priori*, the number of matches for each matrix is unknown and indeed variable across promoters. This introduces an issue for most established classifier learning schemes, as variable length instances are not supported. It is possible to place an upper bound on the number of matches to a given matrix — one more than the difference between the length of the motif and the length of the sequence being searched. This introduces a new problem though. Such a representation produces instances of extremely high (albeit fixed) dimensionality. With this increased dimensionality, a greater amount of training instances are needed to facilitate the learning of salient concepts from the data. This is not practical as there are a finite and relatively small number of positive instances available. Rather than pursuing this thread of at best marginal utility, the problem can be reformulated using multiple instance learning.

3 Multiple Instance Learning

Within the classic paradigm of supervised machine learning the learner is provided with the correct answers to a set of training instances. Within this paper a variation on this technique called multiple instance learning [19] is used. The fundamental differences are the organisation of instances and the availability of class information. Rather than a single instance being described by a vector of attribute values and a class value, instances are grouped into bags. Instances no longer have classes, but rather it is the bag which has a class value attached. It has been suggested that the multiple instance representation lies in generality somewhere between the attribute-vector representation commonly found in supervised learning and the relational representation associated with the field of inductive logic programming [20]. The multiple-instance problem is really a generalisation of the classic supervised learning problem. Alternatively, for ease of expression, classic supervised learning is a special case of MIL where each bag contains only a single instance.

Several approaches have been devised to construct classifiers for multiple instance learning problems [19, 21-24]. Xu and Frank [25], later followed by Ray and Craven [26], explored logistic regression methods. They propose two methods for determining the bag-level class probability – one based on the arithmetic mean of the instance level probability estimations, the other on the geometric mean. Their underlying generative model does not assume that the bag class is determined by only a single instance; a point of interest within this work given more consideration later. Multiple instance learning has been successfully applied to several domains including drug activity prediction [19, 23] and scene recognition [22, 27, 28] where it deals well with ambiguity with respect to which part of the image is of interest [29]. However, to our knowledge this is the first time it has been utilised for promoter prediction.

4 Multiple Instance Learning for Promoter Prediction

With the above approach, it is possible to modify the promoter representation to avoid the problem of variable length instances. Instead of a promoter mapping to an instance, it will instead be represented by a bag. Individual instances will be positive matches within the promoter sequence for the PWM of a given transcription factor. Note that conceptually an instance (a PWM match) can now be described with a small, fixed number of attributes. These are the name of the transcription factor in question and the location of the hit within the sequence of the promoter (i.e. an index). Each instance is assigned to a bag. The bag represents the promoter on which the matches occurred. That is, each promoter is represented in the dataset by a single bag and each bag contains all the (putative) TFBSs found within that promoter. In fact, the variability has not been eliminated from the dataset, it has simply moved to a higher level – the number of instances within a bag. Table 1 presents a small sample dataset, described in the ARFF format [30]. It has two putative promoters (p1, a negative exemplar and p2, a positive exemplar). The first (p1) is described by two PWM hits and the second (p2) by five PWM hits.

Table 1. A sample MIL promoter dataset in ARFF format

```
@relation MIL_SAMPLE

@attribute PROMOTER_NAME {p1, p2}
@attribute TF_MAT_NAME {MA0001, MA0002, MA0003}
@attribute HIT_LOCATION numeric
@attribute PROMOTER? {yes, no}

@data
p1, MA0001, 36,  no
p1, MA0003, 124, no
p2, MA0001, 12,  yes
p2, MA0001, 34,  yes
p2, MA0002, 56,  yes
p2, MA0003, 89,  yes
p2, MA0003, 156, yes
```

It is common when presenting the application of MIL to a problem domain to also present results obtained from applying a non-MIL classifier to the dataset to demonstrate the improvement apparent from using MIL [19]. This is not meaningful here since the instances are simply TFBSs. Individually they do not represent a promoter and hence a non-MIL classifier could not be expected to learn promoter-level concepts from them. For this reason, standard classifiers are not considered when discussing the result possible from a multiple instance learning solution to this problem.

The above paragraph exposes a caveat about this application of MIL. Strictly speaking, Dietterich et al. [19] specify that a bag is positive if any instances within the bag are positive, or the bag is negative if none of the instances in the bag are positive. However, the problem formulation that was given above does not quite align with this. Here, a bag contains positive instances (TFBS hits which make this promoter bag a positive) and negatives (TFBS hits which do no impart positivity to the bag). So far this matches with the general description of an MIL problem, however here it is not simply the case that a single positive TFBS hit in a bag makes the bag positive. Rather, a more complicated underlying concept exists: some combination of positive instances makes the bag positive. Xu and Frank [25] introduce the idea that the label of a bag is determined from an equal and independent contribution of all the instances within the bag. However, within this application, it is reasonable to assume that there is some dependence between the binding sites discovered.

5 Materials and Methods

We explore the application of MIL promoter prediction within two contexts. In the first, it is applied independently. A dataset is generated using a segment of chromosome 21 from the human genome containing fifty-six known promoters. For each promoter, 150bp upstream and downstream of the TSS are extracted (i.e. 300bp in total). Non-promoter sequences are also extracted from the same region, also of 300bp in length each and totalling 560 instances.

In the second context, MIL is applied as a post-processing step to putative promoters predicted by another promoter prediction methodology — specifically physicochemical promoter prediction (PCPP) [18]. Here, the full DNA segment is provided to the PCPP layer (note that the PCPP layer is trained on a separate segment of the same chromosome). Two scenarios are considered — passing all instances to the MIL layer or only passing the positive-classified instances. In both cases, the predicted promoter locations (there are 61, 21 correct, 40 incorrect) are taken as TSSs and 300bp windows are extracted around these. When passing all instances, positive instances not correctly identified by the PCPP layer (there are 35) are included as false negatives. True negative instances are generated in the same fashion as above (there are 560). Considering a second level as re-labelling the instances, there are four possible transitions; each type is of interest. The first is a transition from FP to TN (improving accuracy), the second a transition from TP to FN (decreasing accuracy), the third a transition from FN to TP (improving accuracy) and the last being from TN to FP (decreasing accuracy). Table 2 shows the distribution of instances after applying the PCPP layer, but before applying MIL.

Table 2. Distribution of instances before applying MIL

		Actual Class	
		Positive	*Negative*
PCPP Prediction	*Positive*	21	40
	Negative	35	560

Each element of the dataset described above (i.e. each 300bp segment of DNA) is searched for binding sites using 128 position weight matrices from JASPAR [31] — a high-quality, publicly available repository of matrices. The output from this search for binding sites is arranged as per the multiple instance learning paradigm described above. That is to say, fundamentally, each instance is a matrix hit, described by the name of the matrix and an index into the sequence representing where the hit occurred. Instances are assigned to bags representing the promoter from which they were drawn. In addition to a class label, when evaluating the MIL layer as a second step after PCPP, each bag can also be given the prediction made by the physicochemical promoter prediction software. This allows a multiple-classifier approach utilising the prediction of the lower PCPP layer within the MIL layer. An example dataset (demonstrating the application of just the MIL layer without the PCPP prediction) was presented in Table 2.

As a source of classifiers, MILK [32], a toolkit of multiple instance learning algorithms written as an extension to WEKA [30], is used. Some of the algorithms in MILK are not applicable either due to data incompatibilities or excessive runtime. The logistic regression algorithms presented by Xu and Frank [25] are theoretically well suited to this application, capable of handling the data representation, and have reasonable runtime. Hence, the investigation is concentrated on the use of these. Results are presented for the classifiers MILRARITH (Multiple Instance Logistic Regression with Arithmetic Mean), MILRGEOM (Multiple Instance Logistic Regression with Geometric Mean) and MIRBFNetwork (Multiple Instance Radial Basis Function Network). All experiments are performed using stratified ten-fold cross-validation.

The statistical test employed here is the Wilcoxon signed rank test [33] as described by Conover [34]. Observations are the F-measure for each fold before and after the change in classifier (Note that the split of the dataset into folds is equivalent for all such experiments). Improvements in F-measure, sensitivity or PPV are considered statistically significant if the p-value is less than 0.05.

All of the classifier learning schemes mentioned expose the regression ridge parameter. Empirically it was observed that this parameter influences the quality of classifiers produced. To select a value for this, a nested tenfold cross-validation approach was used. The inner cross validation takes the 90% of the original dataset provided for training in each fold of the outer cross-validation and trains the classifier using a range of possible values for the ridge parameter. Each training of the classifier is itself a complete tenfold cross-validation. The F-measure of each resultant classifier is determined and the parameter value which produces the best F-measure is used to train a final classifier for the given fold of the outer cross validation.

6 Results and Discussion

To put the following results in perspective, one must first consider the performance of the physico-chemical promoter prediction software without the MIL augmentation. Before applying the search for TFBSs and MIL, there are 21 true positives, 35 false negatives, 40 false positives and 560 true negatives. This equates to a sensitivity of 0.38 and a positive predictive value of 0.34. Ideally a balance between sensitivity and positive predictive value is desired. To capture this, here we use F-measure (the harmonic mean of sensitivity and positive predictive value) to represent the quality of the classifier produced.

6.1 PCPP and MIL on Positive-Classified Instances

We begin by examining the main contribution of this paper. That is, the MIL layer utilising only the instances predicted as positive by the PCPP layer. As all the true positives and all the false positives are being provided to the MIL layer, the final count of these is simply the count of true positives and false positives produced by the MIL layer. The final count of true negatives and false negatives is the sum of those produced by the PCPP layer (as none are passed to the MIL layer, they cannot be reclassified) and any new true or false negatives produced by the MIL layer (i.e. false positives correctly reclassified or true positives incorrectly reclassified).

It is not possible for a classifier produced from the positive-predictions-only dataset to demonstrate an improvement in sensitivity over just the PCPP layer. This is obvious if one considers that sensitivity is calculated by dividing the number of true positives by the sum of false negatives and true positives. The denominator of this expression is invariant here since a true positive reclassified becomes a false negative and a false negative reclassified becomes a true positive. The numerator however can be decreased but never increased (it is possible to incorrectly change a true positive to a false negative but there are no false negatives provided to the classifier which might be correctly reclassified to true positives). Hence, the upper bound on sensitivity is that which was produced by the PCPP layer, specifically 0.38.

The results of running the three selected classifier learning schemes using only positively classified instances from the PCPP layer are presented in Table 3. The most striking feature is the improvement apparent from the MIRBFNetwork classifier when propagating only positive-classified instances. The results indicate that the PCPP classifier and the MIRBFNetwork classifier are complementary. The MIRBFNetwork is better at separating true from false positives in the PCPP classifications than the other two classifier types, although all show improvement over just the PCPP layer.

Table 3. Classification performance using PCPP-positive instances. Entries marked in bold show a statistically significant improvement over the base-level PCPP classifier.

	MILRARITH	MILRGEOM	MIRBFNetwork
Sensitivity	0.37	0.37	0.37
Positive Predictive Value	**0.42**	**0.43**	**0.91**
F-Measure	**0.39**	**0.40**	**0.52**

6.2 MIL Promoter Prediction Performance in Isolation

Having demonstrated that MIL can be utilised to improve the classification performance of the PCPP approach, it is instructive to consider how well it might function on its own. Table 4 shows the performance of the three selected classifier learning methods operating in isolation.

The results in Table 4 show that each of the classifier learning methods utilised was not capable of matching the PCPP layer performance in isolation. This demonstrates the efficacy of including the lower layer. The F-measure of about 0.25, although low, is competitive with a lot of existing promoter prediction approaches. It is however much less than the PCPP layer was capable of in isolation.

Table 4. Isolated MIL layer. Only the F-measure is presented.

	MILRARITH	MILRGEOM	MIRBFNetwork
F-Measure	0.25	0.25	0.26

6.3 PCPP and MIL on all Instances

Within this section we consider passing all instances from the PCPP layer regardless of whether they were classified as positive or negative. Recall however, that each instance is appended with the PCPP level classification result. The results for running each of the classifiers using this approach are presented in Table 5. Recall that the PCPP layer achieved a sensitivity of 0.34 a PPV of 0.38 and an F-measure of 0.36. Bold entries in Table 5 are significantly better than the PCPP layer performance.

The results are somewhat mixed. Here, MILRGEOM shows a statistically significant improvement in PPV and a corresponding improvement in F-measure. The MIRBFNetwork classifier scores quite poorly on sensitivity and by extension also F-measure due to a large number of negative predictions.

Table 5. Classification performance using all instances. Bold entries signify a statistically significant improvement over the base-level PCPP classifier.

	MILRARITH	MILRGEOM	MIRBFNetwork
Sensitivity	0.37	0.35	0.15
Positive Predictive Value	**0.40**	**0.44**	**1.00**
F-Measure	0.38	**0.39**	0.26

7 Conclusions and Further Work

Within this paper a multiple instance formulation of the promoter prediction problem was introduced and several algorithms were tested. It has been demonstrated that, with an appropriate selection of learning scheme and parameters, promoters can be predicted using multiple instance learning and a representation based on the location of transcription factor binding sites. Furthermore, the classification performance of this and a physico-chemically based promoter prediction procedure can be improved

by arranging the two in a combined classifier. By using a database such as JASPAR, the performance of such an approach can be expected to improve as more experimentally verified binding sites become available.

The most expensive operation is searching raw DNA sequences for binding sites. In general, the application of this approach in isolation is computationally infeasible. However, by considering a multiple classifier system where the MIL layer is only applied to positive-classified instances from the lower PCPP layer, computational requirements can be sufficiently reduced such that the approach becomes practical. Moreover, this multiple classifier system achieves the best performance in terms of F-measure (0.52), improving upon the PCPP layer (which in this scenario achieved an F-measure of 0.36 and in general is capable of an F-measure of approximately 0.40). Putting this in context, the F-measure of 0.52 beats 6 out of the 9 approaches investigated by Bajic et al. [15].

There are potential avenues for reducing the runtime requirements of this MIL approach. Most obviously, one could scan for binding sites using fewer matrices. In line with this, the identification of which matrices produce hits that are used by the classifier would allow biological insight into promoter function. Those matrices which are not important for classification could then be removed allowing for an improvement in runtime performance.

Further to this, there are additional possibilities for combining classifiers. Here we explored only the inclusion of the lower layer prediction as an attribute at the higher layer, but there is extensive research into multiple classifier systems including approaches such as bagging [35], boosting [36] and stacking, which may improve classification performance.

References

1. International Human Genome Sequencing Consortium, Finishing the euchromatic sequence of the human genome. Nature 431, 931–945 (2004)
2. Kondrakhin, Y.V., Kel, A.E., Kolchanov, N.A., Romashchenko, A.G., Milanesi, L.: Eukaryotic promoter recognition by binding sites for transcription factors. Comput. Appl. Biosci. 11, 477–488 (1995)
3. Prestridge, D.S.: Predicting Pol II Promoter Sequences using Transcription Factor Binding Sites. Journal of Molecular Biology 249, 923–932 (1995)
4. Berman, B.P., Nibu, Y., Pfeiffer, B.D., Tomancak, P., Celniker, S.E., Levine, M., Rubin, G.M., Eisen, M.B.: Exploiting transcription factor binding site clustering to identify cis-regulatory modules involved in pattern formation in the Drosophila genome. Proc. Natl. Acad. Sci. 99(2), 757–762 (2002)
5. Frith, M.C., Li, M.C., Weng, Z.: Cluster-Buster: finding dense clusters of motifs in DNA sequences. Nuc. Acids Res. 31(13), 3666–3668 (2003)
6. Kel, A.E., Kolchanov, N.A., Kapitonov, V.V., Ponomarenko, M.P., Likhachev, A.E., Lim, H.A., Milanesi, L.: Computer analysis and recognition of functional sites on the base of oligonucleotide patterns distributions. In: Second International Conference on Bioinformatics, Supercomputing and Complex Genome Analysis, St. Petersburg Beach, Florida, USA (1993)

7. Narang, V., Sung, W., Mittal, A.: Computational modeling of oligonuceotide positional densities for human promoter prediction. Artificial Intelligence in Medicine 35(1-2), 107–119 (2005)
8. Campbell, N.A., Mitchell, L.G., Reece, J.B.: Biology, 5th edn. Benjamin/Cummings Publ. Co., Inc., Menlo Park (1999)
9. Ohler, U.: Promoter Prediction on a Genomic Scale—The Adh Experience. Genome Res. 10(4), 539–542 (2000)
10. Ohler, U., Liao, G.-C., Niemann, H., Rubin, G.M.: Computational analysis of core promoters in the Drosophila genome. Genome Biol. 3(12) (2002)
11. Ohler, U., Niemann, H.: Identification and analysis of eukaryotic promoters: recent computational approaches. Trends Genet. 17(2), 56–60 (2001)
12. Ohler, U., Niemann, H., Liao, G., Rubin, G.: Joint modeling of DNA sequence and physical properties to improve eukaryotic promoter recognition. Bioinformatics 17(Suppl 1), S199–S206 (2001)
13. Fickett, J.W., Hatzigeorgiou, A.G.: Eukaryotic Promoter Recognition. Genome Research 7, 861–878 (1997)
14. Abeel, T., Saeys, Y., Bonnet, E., Rouze, P., Peer, Y.V.D.: Generic eukaryotic core promoter prediction using structural features of DNA. Genome Res. 18(2), 310–323 (2008)
15. Bajic, V.B., Tan, S.L., Suzuki, Y., Sugano, S.: Promoter prediction analysis on the whole human genome. Nature Biotechnology 22, 1467–1473 (2004)
16. Pedersen, A.G., Baldi, P., Chauvin, Y., Brunak, S.: The biology of eukaryotic promoter prediction-a review. Computers and Chemistry 23(3-4), 191–207 (1999)
17. Oppon, J., Hide, W.: A Statistical Model for Prokaryotic Promoter Prediction. Genome Informatics 9, 271–273 (1998)
18. Uren, P., Cameron-Jones, R.M., Sale, A.: Promoter Prediction Using Physico-chemical Properties of DNA. In: The 2nd International Symposium on Computational Life Science. Springer, Cambridge (2006)
19. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the Multiple Instance Problem with Axis-Parallel Rectangles. Artificial Intelligence 89(1-2), 31–71 (1997)
20. Zucker, J.D., Ganascia, J.G.: Changes of representation for efficient learning in structural domains. In: Thirteenth International Conference on Machine Learning. Morgan Kaufmann, Bary (1996)
21. Auer, P.: On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In: The Fourteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco (1997)
22. Maron, O., Lozano-Perez, T.: A Framework for Multiple-Instance Learning. In: Advances in Neural Information Processing Systems. MIT Press, Cambridge (1998)
23. Zhang, Q., Goldman, S.A.: EM-DD: an improved multiple-instance learning technique. Neural Information Processing Systems 14(10) (2001)
24. Zhou, Z.-H., Zhang, M.-L.: Solving multi-instance problems with classifier ensemble based on constructive clustering. Knowledge and Information Systems 11(2), 155–170 (2007)
25. Xu, X., Frank, E.: Logistic Regression and Boosting for Labeled Bags of Instances. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS, vol. 3056, pp. 272–281. Springer, Heidelberg (2004)
26. Ray, S., Craven, M.: Supervised versus multiple instance learning: An empirical comparison. In: The 22nd International Conference on Machine Learning. ACM Press, New York (2005)

27. Maron, O., Ratan, A.L.: Multiple-instance learning for natural scene classification. In: Fifteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco (1998)
28. Zhou, Z.-H., Zhang, M.-L.: Multi-Instance Multi-Label Learning with Application to Scene Classification. In: Advances in Neural Information Processing Systems, vol. 19. MIT Press, Cambridge (2007)
29. Zhang, Q., Goldman, S.A., Yu, W., Fritts, J.E.: Content-Based Image Retrieval Using Multiple-Instance Learning. In: Nineteenth International Conference on Machine Learning, Sydney, Australia (2002)
30. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
31. Sandelin, A., Alkema, W., Engstrom, P., Wasserman, W.W., Lenhard, B.: JASPAR: an open-access database for eukaryotic transcription factor binding profiles. Nucl. Acids Res. 32(suppl_1), D91–D94 (2004)
32. Xu, X.: Statistical learning in multiple instance problems. Unpublished Masters Thesis, University of Waikato (2003)
33. Wilcoxon, F.: Individual Comparisons by Ranking Methods. Biometrics 1, 80–83 (1945)
34. Conover, W.J.: Practical nonparametric statistics. Wiley, Chichester (1980)
35. Breiman, L.: Bagging Predictors. Machine Learning 24(3), 123–140 (1996)
36. Freund, Y.: Boosting a weak learning algorithm by majority. Information and Computation 121(2), 256–285 (1995)

Revisiting Multiple-Instance Learning Via Embedded Instance Selection

James Foulds and Eibe Frank

Department of Computer Science, University of Waikato, New Zealand
{jf47,eibe}@cs.waikato.ac.nz

Abstract. Multiple-Instance Learning via Embedded Instance Selection (MILES) is a recently proposed multiple-instance (MI) classification algorithm that applies a single-instance base learner to a propositionalized version of MI data. However, the original authors consider only one single-instance base learner for the algorithm — the 1-norm SVM. We present an empirical study investigating the efficacy of alternative base learners for MILES, and compare MILES to other MI algorithms. Our results show that boosted decision stumps can in some cases provide better classification accuracy than the 1-norm SVM as a base learner for MILES. Although MILES provides competitive performance when compared to other MI learners, we identify simpler propositionalization methods that require shorter training times while retaining MILES' strong classification performance on the datasets we tested.

1 Introduction

Multiple-instance (MI) learning is an alternative to the traditional supervised learning model in which learning examples are represented by a *bag* (i.e. multiset) of instances instead of a single feature vector. The MI framework was introduced by Dietterich et al. [7] in the context of a drug-activity prediction problem, where each molecule is represented by a bag of feature vectors corresponding to the conformations (shapes) that the molecule can adopt by rotating its internal bonds. In this problem domain the *standard MI assumption* applies: if and only if at least one instance in a bag is positive (i.e. at least one conformation bonds to the target binding site), then that bag is positive (i.e. the molecule will have the desired drug effect).

Dietterich et al. presented algorithms that learn MI concepts for the *musk* drug activity prediction problem by finding a hyper-rectangle to describe the positive region of instance space. Since then, many other MI learning algorithms have been proposed (see, for example, [1], [2], [10], [12], [14], [24], [25], [27], [28], [29], [30]).

Multiple-instance Learning via Embedded Instance Selection (MILES) is a recent MI learning approach presented by Chen et al. [6], which transforms MI data into a propositionalized form, to which a 1-norm support vector machine (SVM) classifier is applied. Chen et al. do not consider alternatives to the 1-norm SVM, but they do mention briefly that other single-instance base learners

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 300–310, 2008.

are possible. In this paper we view the algorithm as a meta-classifier that can wrap around an arbitrary single-instance learner. We present an empirical study of the performance of the MILES algorithm using a variety of single-instance base learners on a diverse set of benchmark datasets. The goal of the study is to compare the relative performance of different base learners for MILES, and to compare MILES to existing MI algorithms, including other propositionalization methods. The paper is structured as follows. In Section 2, we describe the MILES algorithm. Section 3 details the experimental setup, the results of the experiment are given in Section 4, and we conclude in Section 5.

2 MILES

Multiple-Instance Learning via Embedded Instance Selection (MILES) [6] is an approach to MI learning based on the diverse density framework [14]. In contrast to standard diverse density algorithms, it embeds bags into a single-instance feature space. Most earlier diverse density-based methods have used the standard MI assumption mentioned above and further assume the existence of a single target point[1]. Instead, MILES uses a symmetric assumption, where multiple target points are allowed, each of which may be related to either positive or negative bags. Under this assumption, and using the *most-likely-cause* estimator from the diverse density framework, Chen et al. define a measure specifying the probability that a point x is a target point given a bag, regardless of the bag's class label:

$$Pr(x|B_i) \propto s(x, B_i) = \max_j \exp\left(-\frac{\|x_{ij} - x\|^2}{\sigma^2}\right), \tag{1}$$

where x_{ij} are the instances in bag B_i, and σ is a predefined scaling factor. Note that $s(x, B_i)$ can be interpreted as a measure of similarity between a bag and an instance, determined by the instance x and the closest instance in the bag. MILES uses each instance in the training bags as a candidate for a target point. The candidates are represented as features in an instance-based feature space $\mathbb{F}_c$. Each bag in the training set is mapped into $\mathbb{F}_c$ via the mapping

$$m(B_i) = [s(x^1, B_i), s(x^2, B_i), \cdots, s(x^n, B_i)]^T, \tag{2}$$

where $x^i \in C$ is an instance from the set C of all instances in all of the training bags. When the class labels $c \in \Omega$ of the bags are appended, the resulting space $(\mathbb{F}_c|\Omega)$ is a single-instance feature space. The output of a single-instance classifier trained on this data is used to provide bag-level class labels for future data. The pseudocode of the MILES algorithm is provided in Algorithm 1.

Chen et al. used the 1-norm SVM algorithm as the base classifier, due to the sparsity property of the algorithm — it is known to set most feature weights to zero, which effectively performs feature selection — and the fact that the resulting learning problem is usually very high-dimensional. They do not consider

[1] This means that, roughly speaking, a bag is assumed to be positive if at least some of its instances are close to this point.

Algorithm 1. MILES

D = the set of training bags; C = all instances in the bags in D
L = a single-instance base learner; σ = the scaling factor

$train(D)$
F = an empty set of instances
for (every bag $B_i = \{x_{ij} : j = 1, \cdots, n_i\}$ in D) **do**
  $t = MILES\_transform(B_i)$
  $t.setClassLabel(B_i.getClassLabel())$
  $F = F \cup \{t\}$
$L.train(F)$ //Can optionally perform feature selection here also

$MILES\_transform(B)$, $B = \{x_j : j = 1, \cdots, n\}$ a bag
m(B) = an empty instance of dimension $|C|$
for (every instance x^k in C) **do**
  $d = \min_j \|x_j - x^k\|$; the kth element of $m(B)$ is $s(x^k, B) = e^{\frac{-d^2}{\sigma^2}}$
return $m(B)$

$classify(B)$, $B = \{x_j : j = 1, \cdots, n\}$ a test bag
$t = MILES\_transform(B)$; **return** $L.classify(t)$

alternative base learners, however. We investigate the use of alternative base learners for MILES, and compare the algorithm to other MI approaches.

3 Experiment Design

An extensive set of experiments was performed on a number of multi-instance datasets, using a wide range of MI algorithms and single-instance base learners. The experiments were performed using the WEKA workbench [26]. Each algorithm was evaluated on each dataset by 10 times stratified 10-fold cross-validation. Performance was measured using classification accuracy. We tested for significant differences between algorithms using the corrected resampled t-test [17] with significance level $\alpha = 0.05$.

WEKA implementations were used for all MI algorithms and single-instance base learners, with the exception of MILES and the 1-norm SVM, which were implemented specifically for the experiment. Default parameters were used for each algorithm unless otherwise specified. The MI algorithms were *MILES*, *MISMO* (SVM with the MI polynomial kernel [12]), *mi-SVM* [1], *Citation-KNN* [24], *EMDD* [29], *Adaboost + Optimal Ball* [2], *MIBoost* [27] (with the WEKA REPTree decision tree learner, with no automatic pruning but depth-limited to 3 levels, as the base classifier), *MILR* [27] (using the noisy-or model to combine instance-level probabilities), *MIWrapper* [10], and *SimpleMI* [8].[2]

Of particular interest are MIWrapper and SimpleMI, which, similarly to MILES, are wrapper algorithms that apply a single-instance base learner to

[2] Where the algorithm was not explicitly named by the original authors, the name of the WEKA implementation has been used instead.

a propositionalized version of the given MI data. MIWrapper performs propositionalization by applying bag-level class labels to instances, and weighting the instances so that each bag has the same total weight. A single-instance model is built on the resulting dataset, and bag-level predictions are made by averaging the predicted probabilities of instances in a bag. SimpleMI performs propositionalization by averaging the attribute values of the instances in each bag, and appending the bag's class label to the resulting feature vector. The wrapper algorithms were evaluated using the following single-instance base learners: *C4.5* [20], *random forests* (100 trees) [5], *Adaboost* [11] + *C4.5* (10 iterations), *Adaboost* + *decision stumps* (1-level decision trees, 100 iterations), *bagging* [4] + *C4.5* (10 iterations), *2-norm SVMs* (SMO) [19] with a linear kernel and a radial-basis function kernel, the linear *1-norm SVM*, and *logistic regression*.

The datasets are described very briefly here, with the names of the datasets (as labeled in the results tables) italicized for convenience. The *musk1* and *musk2* datasets are the musk data used in [7]. Each bag represents a molecule, and the task is to predict whether the molecule emits a musky odour. *Eastwest* is the train direction prediction problem from the East West Challenge ILP contest [16]. *Westeast* is exactly the same problem as *eastwest*, except that the class labels are reversed. This is an interesting variation because *eastwest* is compatible with the standard MI assumption, while *westeast* is not [8].

The mutagenicity prediction problem [22] was also used in the experiments. Three representations proposed by [21] for transforming the mutagenesis ILP problem into a multi-instance problem were used, which were labeled *muta-atoms*, *muta-bonds* and *muta-chains*. The *suramin* dataset [3] is another ILP-based drug activity prediction problem, where the task is to detect suramin analogues that can act as anti-cancer agents. The *thioredoxin* dataset is the thioredoxin-fold protein identification task proposed by [23].

Two sets of image data for Content-based Image Retrieval (CBIR) tasks were used, each containing three different image categories. These image databases provided six different image retrieval problems — one for each image category, with the task being to identify images belonging to the target category. The first image database was originally provided by [1], and contains MI bags representing photographs of *elephants*, *foxes* and *tigers* from the Corel dataset. The second CBIR dataset was the GRAZ02 [18] dataset, containing images of *bikes*, *cars* and *people*, with features derived from the Ohta colour space representations of the image as in [15].

4 Experimental Results and Analysis

This section presents a comparison of base learners for MILES, and compares the algorithm with other MI learning methods. The reader is referred to the first author's MSc thesis for more detailed experimental results [9].

Given the number of algorithms and datasets investigated, parameter tuning for all of the MI algorithms and base classifiers was infeasible. However, with

the exception of the SVMs, the classification schemes used in the experiment had fairly robust parameter values already provided by the default settings in their WEKA implementations. We set the scaling parameter for MILES to $\sigma^2 = 8 \times 10^5$, as used by Chen et al. for *musk2*. As we found that the value for the 1-norm SVM regularization parameter selected by Chen et al. for *musk2* ($\lambda = 0.45$) produced poor results for MILES on many of the datasets, we performed internal cross-validation to select the best λ value for each fold. We evaluated six candidate values of λ via two-fold cross-validation on the training data for each fold of the ten repeats of ten-fold cross-validation, selecting the value which produced the highest classification accuracy. Finally, a greedy search was performed by iteratively evaluating adjacent candidates to the currently selected value via ten-fold cross-validation (again, on the training data of the fold). This internal cross-validation parameter search was performed using the GridSearch algorithm in WEKA. The candidate values were powers of ten between 10^{-1} and 10^{-6}. The same internal cross-validation method was also used to select the C regularization parameter for the 2-norm SVMs, with candidate values being powers of ten between 10^3 and 10^{-2}.

4.1 Comparison of Base Learners for MILES

A major goal of the experiment was to compare different base learners for MILES, particularly with respect to the 1-norm SVM. The results of this part of the experiment are displayed in Tables 1 and 2.

Table 1. MILES: Percentage Accuracy for Non-Ensemble Base Learners

Dataset	1-Norm SVM	C4.5	Logistic Regression	SMO (LIN)	SMO (RBF)
musk1	83.3±11.8	84.1±11.9	84.8±11.3	86.9±10.4	89.1±10.1
musk2	91.6±8.3	82.5±12.1 •	85.8±11.0	88.4±9.7	79.5±12.5 •
eastwest	74.0±25.1	50.0±0.0 •	64.5±29.6	55.5±30.9	55.5±31.7
westeast	74.0±25.1	50.0±0.0 •	68.5±33.1	54.0±30.7	54.5±31.1
muta-atoms	74.8±14.4	80.8±8.1	83.8±7.2	80.8±8.8	83.7±9.2
muta-bonds	72.2±12.7	77.1±9.8	80.2±8.8	79.8±9.5	81.8±8.9 ○
muta-chains	75.9±9.2	79.3±9.5	73.5±9.4	77.9±8.5	78.6±10.3
suramin	65.0±45.2	65.0±45.2	65.0±45.2	65.0±45.2	65.0±45.2
thioredoxin	88.1±5.1	84.3±7.1	87.1±3.9*	69.1±10.3 •	86.3±4.3
elephant	84.1±8.9	77.5±9.2	79.6±9.1	83.9±9.0	83.4±8.9
fox	63.0±9.5	56.8±11.2	63.6±8.9	64.8±9.5	64.2±9.7
tiger	80.7±8.3	69.7±9.3 •	80.0±9.2	81.5±8.4	81.7±8.9
bikes	78.4±4.2	72.5±5.7 •	72.4±4.8 •	80.1±4.9	78.7±4.9
cars	72.2±4.3	62.6±4.7 •	63.9±4.9 •	72.0±4.7	71.9±4.7
people	74.4±5.0	69.8±5.8 •	66.9±5.0 •	74.3±4.8	75.9±4.8

○, • statistically significant improvement or degradation vs 1-norm SVM

∗ Thioredoxin result obtained using the *SimpleLogistic* [13] implementation in WEKA, due to memory problems with *Logistic*.

Table 2. MILES: Percentage Accuracy for Ensemble Base Learners

Dataset	1-Norm SVM	Adaboost + D. Stump	Random Forest	Adaboost + C4.5	Bagging + C4.5
musk1	83.3±11.8	88.0±11.6	87.0±11.4	85.8±12.0	86.0±11.5
musk2	91.6±8.3	83.2±11.5 •	81.7±11.2 •	83.2±11.3 •	83.7±11.5 •
eastwest	74.0±25.1	81.0±24.4	80.0±24.6	50.0±0.0 •	50.5±5.0 •
westeast	74.0±25.1	81.0±24.4	80.0±24.6	50.0±0.0 •	50.5±5.0 •
muta-atoms	74.8±14.4	83.9±8.6	82.0±8.2	79.5±8.5	80.5±7.7
muta-bonds	72.2±12.7	86.3±7.4 ○	79.7±10.5	80.1±9.9	77.4±8.9
muta-chains	75.9±9.2	86.0±8.0 ○	80.4±9.2	80.8±8.1	79.8±9.1
suramin	65.0±45.2	65.0±45.2	65.0±45.2	65.0±45.2	62.0±46.1
thioredoxin	88.1±5.1	89.3±4.0	87.7±2.7	85.6±6.4	88.2±4.6
elephant	84.1±8.9	80.9±7.7	82.3±8.2	81.5±8.9	84.0±8.3
fox	63.0±9.5	61.6±10.9	64.9±10.2	59.4±11.6	61.4±10.3
tiger	80.7±8.3	80.5±8.9	78.6±9.0	75.4±9.3	75.7±8.4
bikes	78.4±4.2	78.0±5.0	79.2±4.4	78.0±4.5	77.7±5.1
cars	72.2±4.3	71.6±4.1	71.7±4.0	69.3±5.0	70.5±4.9
people	74.4±5.0	75.6±4.6	77.5±4.3	75.4±4.8	76.6±4.7

○, • statistically significant improvement or degradation vs 1-norm SVM

As the results show, the 1-norm SVM was competitive against the other base learners, with no other base learner consistently performing significantly better than it. However, Adaboost with decision stumps had two significant wins and only one significant loss versus the 1-norm SVM (Table 2).

The 2-norm SVMs with the linear and RBF kernels were both very competitive with the 1-norm method. The 2-norm SVM with the RBF kernel had one significant win and one significant loss versus the 1-norm SVM, while the 2-norm SVM with the linear kernel was only significantly worse than the 1-norm SVM on the *thieoredoxin* dataset (Table 1). These results indicate that the 1-norm SVM does not have a clear advantage over its 2-norm cousin as a base learner for MILES. In the observed experimental results, the increased sparsity of the 1-norm SVM did not translate into consistently superior classification accuracy, despite the high dimensionality of the datasets produced by the MILES transformation. However, the 1-norm SVM did outperform logistic regression, which produces linear models that do not exhibit any sparsity (Table 1).

The *eastwest* and *westeast* datasets were problematic for many MILES base learners, with half of the schemes performing little or no better than chance on these problems, although several schemes achieved accuracies of around 80%. Note that the results were similar or identical for both datasets, regardless of the base learner. This is as expected, given that MILES is designed to use a symmetric MI assumption.

MILES' performance was consistent on the suramin problem. All base learning schemes achieved an accuracy of 65.0% on this dataset, except for bagging with C4.5, where an accuracy value of 62.0% was observed (Table 2). The small size of the dataset at least partially explains the consistency between

schemes — it contains only 11 bags, albeit with many instances in each of those bags.

Random forests and Adaboost.M1 with decison stumps were the standout ensemble base learners for MILES (see Table 2). These classifiers only performed significantly worse than the 1-norm SVM on *musk2*. Furthermore, boosted decision stumps had the highest accuracy for any MILES base learner tried in the experiments on the *eastwest / westeast* datasets, all three *mutagenesis* datasets, *thieoredoxin*, and also matched the performance of the other base learners on *suramin*. Two of these results were significantly superior to the 1-norm SVM. Boosted decision stumps had slightly lower accuracies than the 1-norm SVM on five of the six image datasets, but these differences were not statistically significant. It should also be noted that parameter tuning with cross-validation was not necessary to achieve good results using Adaboost with decision stumps, unlike for the 1-norm SVM.

Although single C4.5 trees perform poorly (see Table 1), the results also show that boosted and bagged C4.5 trees perform well. However, boosted and bagged C4.5 performed no better than chance on *eastwest* and *westeast* and consequently suffered signficant losses against the 1-norm SVM on those datasets.

The strong performance of Adaboost.M1 with decision stumps is interesting, given the relationship between this model and the SVM model recommended by Chen et al (2006). Like support vector machines, the hypothesis learnt by Adaboost.M1 is a weighted linear threshold. When decision stumps are used, each weak learner corresponds to an attribute (i.e. the attribute that the decision stump splits on), and the weights for the weak learners perform a similar function to the attribute weights learnt by a linear SVM. As in SVMs, the solution is sparse because only a subset of the attributes is selected into the ensemble.

4.2 Comparison of MILES to Other Wrapper Algorithms

In this section we compare MILES to the two other wrapper algorithms — SimpleMI and MIWrapper — with respect to classification accuracy and training time, using Adaboost with decision stumps (100 stumps) as the base learner. Table 3 shows the classification accuracy and training time results for the algorithms.

The results show that in most cases all three propositionalization schemes give similar classification performance. There were no significant differences between MILES and SimpleMI for classification accuracy using this base learner. MILES was superior to MIWrapper on the *mutagenesis* datasets, but MIWrapper had significantly higher accuracy on the *people* dataset.

The results also show that there are substantial differences in training time. SimpleMI always had the shortest training time of the three methods for all datasets, almost always followed by MIWrapper, with MILES being the slowest of the wrapper algorithms on all datasets except *musk1*. This is unsurprising, given that the SimpleMI method only generates one instance for each training bag, without increasing the dimensionality of the feature space. Although MILES also generates one instance per training bag, the dimensionality of the feature

Table 3. Comparison of Wrapper Algorithms using Adaboost with Decision Stump Base Learner (100 Stumps)

	Percentage Accuracy			Training time (CPU Seconds)		
Dataset	MILES	SimpleMI	MIWrapper	MILES	SimpleMI	MIWrapper
musk1	88.0±11.6	83.2±12.3	84.7±10.7	4.3±0.3	3.0±0.1 ○	4.7±0.2 ●
musk2	83.2±11.5	78.7±11.9	79.7±10.6	284.7±41.5	3.5±0.0 ○	151.8±19.0 ○
eastwest	81.0±24.4	80.0±31.8	69.0±26.4	0.2±0.1	0.0±0.0 ○	0.2±0.0
westeast	81.0±24.4	81.5±31.5	69.0±26.4	0.2±0.1	0.0±0.0 ○	0.2±0.0
muta-atoms	83.9±8.6	80.3±8.4	66.5±2.3 ●	17.9±0.2	0.1±0.0 ○	0.7±0.0 ○
muta-bonds	86.3±7.4	85.8±7.7	73.2±8.4 ●	53.5±0.6	0.2±0.0 ○	3.1±0.1 ○
muta-chains	86.0±8.0	81.2±8.8	74.0±7.7 ●	88.1±0.8	0.2±0.0 ○	10.2±0.2 ○
suramin	65.0±45.2	53.0±48.1	65.0±45.2	3.7±0.3	0.0±0.0 ○	1.7±0.1 ○
thioredoxin	89.3±4.0	86.2±5.3	87.1±2.4	624.2±6.7	0.1±0.0 ○	32.2±0.4 ○
elephant	80.9±7.7	86.5±8.1	85.5±7.3	36.5±0.4	3.7±0.1 ○	15.8±0.2 ○
fox	61.6±10.9	67.0±10.5	65.7±9.6	34.0±0.4	3.3±0.9 ○	14.7±0.1 ○
tiger	80.5±8.9	82.5±8.7	81.8±8.5	30.4±0.4	1.8±0.3 ○	13.7±0.1 ○
bikes	78.0±5.0	80.3±4.9	79.2±4.6	451.2±1.3	12.3±0.2 ○	66.3±0.3 ○
cars	71.6±4.1	74.4±4.4	71.3±4.8	524.7±0.5	5.2±0.0 ○	73.4±0.3 ○
people	75.6±4.6	79.0±4.8	79.5±4.3 ○	385.1±0.4	4.5±0.0 ○	58.7±0.2 ○

○, ● statistically significant improvement or degradation vs MILES

space is almost always much higher, as the number of attributes is equal to the total number of instances in the training bags. In contrast, MIWrapper generates one instance for every instance in every bag, leaving the dimensionality of the feature space unchanged.

4.3 Overall Comparison of Classification Accuracy

The classification accuracy of the best variants of the three wrapper schemes MILES, MIWrapper, and SimpleMI, as well as the accuracy of the best of the other MI algorithms listed in Section 3 are shown in Table 4. Interestingly, the best results for each type of scheme were seldom more than a few percentage points different from each other. Notable exceptions to this are the *eastwest* / *westeast* datasets, where the best MILES classifier was around ten percentage points ahead of the best MIWrapper classifier and the best non-wrapper scheme, and SimpleMI was fourteen percentage points ahead of the best MILES scheme. On the *suramin* dataset, MIWrapper with the linear SVM base learner achieved an accuracy of 95%, which was 30-40 percentage points ahead of all other schemes. However, this result was not statistically superior to the majority of the other schemes, possibly due to the small size of the dataset (11 bags). There was also a difference of around eight percentage points between the best MILES classifier and the best SimpleMI and MIWrapper classifiers on the *musk2* dataset. Note that the σ value for MILES used in these experiments was selected by [6] based on tuning experiments on a subset of the *musk2* dataset, so the results for MILES on that dataset may be optimistic.

Table 4. The Best Result For Each Type of Scheme

Dataset	Best MILES	%	Best MIWrapper	%	Best SimpleMI	%	Best Other MI Learners	%
musk1	SMO (RBF)	89.1	Random Forest	87.3	SMO (RBF)	86.2	EMDD	85.2
musk2	1-Norm SVM	91.6	SMO (RBF)	83.0	SMO (RBF)	83.8	EMDD	84.7
eastwest	Adaboost + D.Stump	81.0	Adaboost + D. Stump	69.0	C4.5	95.0	Adaboost + Opt.Ball	71.5
westeast	Adaboost + D.Stump	81.0	Adaboost + D. Stump	69.0	C4.5	95.0	MISMO	70.0
muta-atom	Adaboost + D. Stump	83.9	Random Forest	81.9	Random Forest	80.9	MIBOOST + REPTree	77.8
muta-bond	Adaboost + D. Stump	86.3	Random Forest	83.1	Random* Forest	85.8	MIBOOST + REPTree	84.4
muta-chains	Adaboost + D. Stump	86.0	Bagging + C4.5	85.3	Random Forest	83.5	MIBOOST + REPTree	82.3
suramin	1-Norm* SVM	65.0	SMO (LIN)	95.0	SMO (LIN)	74.0	Citation* KNN	65.0
thioredoxin	Adaboost + D. Stump	89.3	Adaboost + C4.5	88.0	Logistic Regression	87.6	Adaboost + Opt.Ball	90.3
elephant	1-Norm SVM	84.1	Random Forest	87.1	Random Forest	87.3	MIBOOST + REPTree	82.8
fox	Random Forest	64.9	Adaboost* + D. Stump	65.7	Adaboost +D.Stump	67.0	MIBOOST + REPTree	66.3
tiger	SMO (RBF)	81.7	Random Forest	84.3	Random Forest	82.9	MIBOOST + REPTree	82.2
bikes	SMO (LIN)	80.1	SMO (RBF)	83.2	1-Norm SVM	84.3	mi-SVM	83.5
cars	1-Norm SVM	72.2	Random Forest	74.8	Random Forest	76.5	Adaboost + Opt.Ball	72.2
people	Random Forest	77.5	Random Forest	82.6	Random Forest	81.5	MIBOOST + REPTree	78.9

∗ Scheme was best-equal with one or more other schemes.

As mentioned previously, Adaboost with decision stumps was the dominant base learner for MILES, being the best (or best-equal) scheme for six of the fifteen datasets. MIBoost was the strongest overall method of the other MI algorithms, and the random forests algorithm was the best overall base learner for MIWrapper and SimpleMI.

5 Conclusions

The goals of the study were to compare base learners for MILES, and to compare MILES to other state-of-the-art MI algorithms. The results indicate that the 1-norm SVM is not generally superior to the standard 2-norm SVM as a base learner

for MILES, despite the sparsity property that was thought to be important for the high-dimensional feature space created by the MILES transformation [6]. Moreover, although the 1-norm SVM was a competitive base learner for MILES in the experiment, Adaboost with decision stumps exhibited higher classification accuracy for some problem domains and did not require parameter tuning.

The results also show that when appropriate base learners are used, MILES is competitive in classification performance with any MI algorithm we considered. However, the simpler MIWrapper and SimpleMI methods almost always perform just as well as MILES, despite being significantly superior in terms of CPU training time. To achieve good classification accuracy in a wide variety of cases, random forests can be recommended as a base learner for MIWrapper and SimpleMI.

Perhaps the most interesting result of the experiments is the effectiveness of the extremely simple propositionalization methods SimpleMI and MIWrapper in comparison to MILES. The results also confirm their good performance when compared to more sophisticated dedicated MI algorithms (see also [8]). It appears to be an open problem to find MI algorithms that are superior to these simple propositionalization techniques on benchmark datasets, or to find problems where dedicated MI algorithms are more effective than propositionalization.

In future work, it would be interesting to compare MILES, SimpleMI and MIWrapper to other multi-instance propositionalization methods such as TLC [25], and the recent CCE [30] and BARTMIP [28] algorithms. In particular, BARTMIP is an algorithm that is closely related to MILES, where propositionalization is performed based on distances from *bags*, rather than distances from *points* as in the latter algorithm, so a thorough comparison of those two algorithms would be particularly insightful.

Acknowledgement

We would like to thank Michael Mayo for providing us with the multi-instance version of the GRAZ02 dataset.

References

1. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: NIPS, pp. 577–584 (2002)
2. Auer, P., Ortner, R.: A boosting approach to multiple instance learning. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 63–74. Springer, Heidelberg (2004)
3. Braddock, P.S., Hu, D.E., Fan, T.P., Stratford, I.J., Harris, A.L., Bicknell, R.: A structure-activity analysis of antagonism of the growth factor and angiogenic activity of basic fibroblast growth factor by suramin and related polyanions. Br. J. Cancer 69(5), 890–898 (1994)
4. Breiman, L.: Bagging predictors. ML 24(2), 123–140 (1996)
5. Breiman, L.: Random forests. ML 45(1), 5–32 (2001)
6. Chen, Y., Bi, J., Wang, J.Z.: MILES: Multiple-instance learning via embedded instance selection. IEEE PAMI 28(12), 1931–1947 (2006)
7. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. AI 89(1-2), 31–71 (1997)

8. Dong, L.: A comparison of multi-instance learning algorithms. Master's thesis, University of Waikato (2006)
9. Foulds, J.: Learning instance weights in multi-instance learning. Master's thesis, University of Waikato (2008)
10. Frank, E., Xu, X.: Applying propositional learning algorithms to multi-instance data. Technical report, Dept. of Computer Science, University of Waikato (2003)
11. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: ICML, pp. 148–156 (1996)
12. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.J.: Multi-instance kernels. In: ICML, pp. 179–186 (2002)
13. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 241–252. Springer, Heidelberg (2003)
14. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: NIPS (1997)
15. Mayo, M.: Effective classifiers for detecting objects. In: CIRAS (2007)
16. Michie, D., Muggleton, S., Page, D., Srinivasan, A.: A new East-West challenge. Technical report, Oxford University Computing Laboratory (1994)
17. Nadeau, C., Bengio, Y.: Inference for the Generalization Error. ML 52(3), 239–281 (2003)
18. Opelt, A., Pinz, A., Fussenegger, M., Auer, P.: Generic object recognition with boosting. IEEE PAMI 28(3), 416–431 (2006)
19. Platt, J.: Fast training of support vector machines using sequential minimal optimization. Advances in kernel methods: support vector learning, 185–208 (1999)
20. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
21. Reutemann, P.: Development of a propositionalization toolbox. Master's thesis, Albert Ludwigs University of Freiburg (2004)
22. Srinivasan, A., Muggleton, S., King, R.D., Sternberg, M.J.E.: Mutagenesis: ILP experiments in a non-determinate biological domain. In: ILP, pp. 217–232 (1994)
23. Wang, C., Scott, S.D., Zhang, J., Tao, Q., Fomenko, D., Gladyshev, V.: A study in modeling low-conservation protein superfamilies. Technical report, Dept. of Comp. Sci., University of Nebraska-Lincoln (2004)
24. Wang, J., Zucker, J.-D.: Solving the multiple-instance problem: A lazy learning approach. In: ICML, pp. 1119–1125 (2000)
25. Weidmann, N., Frank, E., Pfahringer, B.: A two-level learning method for generalized multi-instance problems. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 468–479. Springer, Heidelberg (2003)
26. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
27. Xu, X., Frank, E.: Logistic regression and boosting for labeled bags of instances. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 272–281. Springer, Heidelberg (2004)
28. Zhang, M.-L., Zhou, Z.-H.: Multi-instance clustering with applications to multi-instance prediction. Applied Intelligence (in press)
29. Zhang, Q., Goldman, S.: EM-DD: An improved multiple-instance learning technique. In: NIPS, pp. 1073–1080 (2001)
30. Zhou, Z.-H., Zhang, M.-L.: Solving multi-instance problems with classifier ensemble based on constructive clustering. KAIS 11(2), 155–170 (2007)

Decision Tree Induction from Numeric Data Stream

Satoru Nishimura, Masahiro Terabe, and Kazuo Hashimoto

Graduate School of Information Sciences, Tohoku University,
6-6-11-304 Aramaki-Aza-Aoba, Aoba-Ku, Sendai, Miyagi, 980-8579, Japan
{nishimura,terabe,kh}@aiet.ecei.tohoku.ac.jp

Abstract. Hoeffding Tree Algorithm is known as a method to induce decision trees from a data stream. Treatment of numeric attribute on Hoeffding Tree Algorithm has been discussed for stationary input. It has not yet investigated, however, for non-stationary input where the effect of concept drift is apparent. This paper identifies three major approaches to handle numeric values, Exhaustive Method, Gaussian Approximation, and Discretizaion Method, and through experiment shows the best suited modeling of numeric attributes for Hoeffding Tree Algorithm. This paper also experimentaly compares the performance of two known methods for concept drift detection, Hoeffding Bound Based Method and Accuracy Based Method.

Keywords: Numeric Data Stream, Concept Drift, Hoeffding Tree.

1 Introduction

Due to the recent development of sensor technologies and spread of the Internet, knowledge discovery from data stream becomes major concern in many applications. Data stream has the characteristics such that (1) data arrive continuously, (2) concept drift may take place in the course of time. It is necessary to develop a new learning algorithm or method suitable for the above characteristics of data stream. This paper discusses a decision tree induction from data stream under concept drift.

Decision tree Based Method is well accepted because of its lightness in computation and easiness in understanding of the result, Hoeffding Tree Algorithm [1,2,3,4,5] (HTA hereafter) is a method to induce decision trees fast from a data stream. Some HTAs adopt naive-Bayes Classifiers (NBCs hereafter) in the leaf nodes of a decision tree. For example, we proposed CVFDT$_{\mathrm{NBC}}$ [5]. CVFDT$_{\mathrm{NBC}}$ adopts NBCs in the leaf nodes of a decision tree induced by CVFDT. We presented that CVFDT$_{\mathrm{NBC}}$ can induce higher accuracy classifier than CVFDT does.

To the best of our knowledge, only Pfahringer et al. [6] discusses suitable modeling of numeric attributes in HTA. In terms of accuracy of decision trees and memory consumption, they proved that assumption of Gaussian distribution is suitable for HTA without NBCs in stationary environment.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 311–317, 2008.

In stationary environment, only fitness of numeric attribute's model to actual data affects accuracy. In non-stationary environment, tracking ability to concept drift, updating numeric attribute's model, affects accuracy of decision trees as well as fitness. Suitable modeling of numeric attributes in non-stationary environment could be different from the one in stationary environment. Moreover, adopting NBCs to HTA could affect suitable modeling of numeric attribute. This paper, using a major HTA of CVFDT and its extended version of $\text{CVFDT}_{\text{NBC}}$, discusses the most suitable modeling of numeric attribute to CVFDT and $\text{CVFDT}_{\text{NBC}}$ in terms of accuracy of decision tree, processing time of decision tree induction, and memory consumption.

As concept drift degrades the accuracy, detecting concept drift is crucial to keep accurate decision tree. Although various concept drift detection method, using Hoeffding Bound or using accuracy of classifiers, are proposed. There is no evaluation between concept drift detection methods. Using various data streams with different speed of concept drift, this paper compares the conventional method using Hoeffding Bound and the one using Accuracy proposed by Gama et al.

2 The Features of Major Hoeffding Tree Algorithm

VFDT [1], CVFDT [4], $\text{CVFDT}_{\text{NBC}}$ [5], and UFFT [3] are HTAs. The features of each methods is listed in Table 1.

Table 1. The Features of Major Hoeffding Tree Algorithm

Features	Classification Method	Tree Type	Numeric Attributes	Drift Detection
VFDT	Majority Class	Single Tree	Discretization	(can't detect)
CVFDT			Exaustive Method	Hoeffding Bound
$\text{CVFDT}_{\text{NBC}}$	naive Bayes		(can't handle)	
UFFT		Ensemble	Gaussian Approx.	Accuracy Change

VFDT, CVFDT, and $\text{CVFDT}_{\text{NBC}}$ induce a single decision tree. UFFT, however, induces decision trees and constructs an ensemble of decision trees. VFDT and CVFDT classify examples based on the majority class of leaf nodes, while other methods classify examples using NBCs at leaf nodes.

VFDT discretizes numeric attributes and stores joint frequencies of each discretized interval (Discretization Method). VFDT's discretization, however, contains the problem such that the order of examples affects performance of discretization. CVFDT stores all the numeric attribute values, and splits a leaf node based on the attribute value that maximizes the information gain (Exhaustive Method). $\text{CVFDT}_{\text{NBC}}$ calculates a set of conditional probability by discretizing numeric attributes to feed them into NBCs, since the current $\text{CVFDT}_{\text{NBC}}$ is unable to handle numeric attributes. UFFT, on the other hand, models numeric attributes by assuming Gaussian distribution (Gaussian Approximation).

VFDT supposes stationary data stream and do not equip concept drift detection method. CVFDT and $\text{CVFDT}_{\text{NBC}}$ periodically check best split attributes at all nodes of a decision tree by Hoeffding Bound, and detect concept drift (Hoeffding Bound Based Method). But, UFFT detects concept drift by observing the accuracy degradation at each node.

3 Criteria for Modeling Numeric Attributes

This paper compares 3 possible modeling of numeric attributes for the decision tree induction on HTA; Exhaustive Method, Gaussian Approximation, Discretization Method. Since VFDT's discretization is unstable depending on the order of example's arrival, this paper, by adopting Entropy-Based Discretization [7], discretizes numeric attributes into binary intervals based on the attribute value that maximizes information gain. There should be a certain upper limit of examples to be stored at each leaf node to discretize numeric attribute. Although there should be a mechanism to decide a suitable number of examples, it is outside the scope of this paper. This paper simply introduces an user defined number d, for the number of examples, where d is assumed to be 500 in this paper.

3.1 Fitness to Acutual Data

Fitness of numeric attribute's model affects accuracy of decision trees induced by HTA . Exhaustive Method stores all the attribute values at each node to exactly calculate the threshold for node split. However, it takes huge amount of processing time compared to other methods. Gaussian Approximation updates only the limited number of parameters of Gaussian distribution each arrival of example. The fast processing is guaranteed due to the drastic reduction in the amount of information necessary for calculation. When acutual distribution differs from Gaussian distribution, Gaussian Approximation would split leaf nodes unappropriately. Discretization Method reduces the amount of information stored in a decision tree, since it is not necessary to store attribute value after discretization. Discretization Method is able to split leaf nodes correctly, since it splits leaf nodes based on acutual examples.

3.2 Tracking Ability to Concept Drift

We have to detect concept drift by feature value that is calculated based on numeric attribute's model, tracking ability of numeric attribute's model to concept drift is important. In this paper, we consider Hoeffding Bound Based Method and Accuracy Based Method, we discuss effect of numeric attribute's model to these two method. Hoeffding Bound Based Method needs calculation of information gain. Exhaustive Method could calculate exact information gain. Other methods, however, are difficult to calculate exact information gain, because other methods don't hold attribute values, they are weak to concept drift. Accuracy Based Method, on the other hand, needs calculation of accuracy of classifiers. If

concept drift happens, accuracy degrades independent from numeric attribute's model, even when numeric attribute's model differs from appropriate one. Accuracy Based Method is independent from numeric attribute's model.

3.3 Influences of Each Method to Naive Bayes Classifier

In the case of HTA with NBCs, we have to consider the effect of numeric attribute's model for running NBCs. To calculate probability that is needed for NBCs, Exhaustive Method have to store numeric attribute values in special form as in $VFDT_C$ or discretize numeric attributes before running NBCs. Gaussian Approximation, on the other hand, could easily calculate probability to run NBCs. Discretization Method discretizes numeric attributes after storing d examples. Since Discretization Method is unable to run NBCs before storing d examples, accuracy of classifier would degrade until d examples are stored.

4 Evaluation of Concept Drift Detection Methods

Various concept drift detection methods, using Hoeffding Bound and using accuracy of classifiers, are proposed. The features of each method is shown below.

4.1 Hoeffding Bound Based Method

To detect concept drift, CVFDT and $\text{CVFDT}_{\text{NBC}}$ periodically check the best split attributes at all nodes of a decision tree by Hoeffding Bound. If concept drift is detected at a node, CVFDT and $\text{CVFDT}_{\text{NBC}}$ start to generate an alternative subtree for an old subtree whose root node is the node. The alternative subtree is replaced with the old subtree when it becomes more accurate than the old one.

Concept drift degrades the accuracy of classifiers, especially when an abrupt and big drift takes place. Decision tree should be renewed at every occurrence of concept drift to keep the accuracy by reflecting the latest statistical nature of data stream. In the case of abrupt concept drift, it is necessary to generate an alternative subtree near the root node. Since nodes close to the root node store many examples, it is difficult to detect concept drift quickly by the influence of old concept's examples.

4.2 Accuracy Based Method

In this paper, we introduce to CVFDT and $\text{CVFDT}_{\text{NBC}}$ the method used in UFFT [3] (Accuracy Based Method) with a modification to single decision tree learner.

During the induction of a decision tree, each node classifies examples that traversed the node. Monitoring the change of classifier's accuracy, Accuracy Based Method detects concept drift. If abrupt concept drift happens, accuracy of classifiers would degrade extremely. Accuracy Based Method can detect concept drift more quickly than Hoeffding Bound Based Method does. Each arrival of example, Accuracy Based Method, however, takes extra time to classify an example at each node that the example traversed.

If UFFT detects concept drift at some nodes, UFFT prunes the subtrees whose root node are those nodes. By constructing an ensemble of decision trees, UFFT avoids radical pruning of decision trees. Single decision tree learners such as CVFDT and $\text{CVFDT}_{\text{NBC}}$ would suffer from degradation of accuracy caused by radical pruning of a decision tree. Such a case is safely avoided by waiting for the growth of alternative subtree until it becomes more accurate than the old one.

5 Experiments

This section clarifies through experiments how the modeling of numeric attribute and the concept drift detection method affects CVFDT and $\text{CVFDT}_{\text{NBC}}$ in terms of accuracy, processing time and memory consumpstion for decision tree induction. Following suggestions shown in Hulten et al. [4], we generated artificial numeric data streams, which consist of 3,000,000 of examples, with concept drift to carry out performance tests. Whenever every 10,000 examples are streamed in, we evaluate the accuracy of the induced decision tree and processing time for decision tree induction, by giving 10,000 test examples which represent the latest concept. Therefore, we will conduct 300 tests, and the accuracy, the processing time for learning 10,000 examples are given by the average of 300 tests. Concept drift is generated for every 50,000 examples. The parameters of CVFDT is the same as Hulten reported.

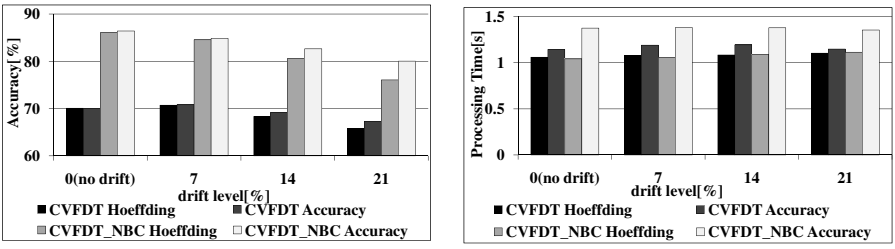
5.1 Experimental Results

(1) Effects of Numeric Attribute Treatment. Accuracy, processing time, and memory consumption are measured for CVFDT and $\text{CVFDT}_{\text{NBC}}$ with different modeling of numeric attribute. The number of attribute is 10 in the generated data stream, with about 7% of example classes are reversed as concept drift at every 50,000 examples. Table 2 shows performances of each method. Bold number shows the statistical difference by 5% of significance level. $\text{CVFDT}_{\text{NBC}}$ is measured with only Gaussian Approximation and Discretization Method, since Exhaustive Method needs extra process of calculating the probability.

Table 2 shows that accuracy of CVFDT is independent of the modeling of numeric attribute, except that Exhaustive Method shows about 0.15% higher accuracy. On the other hand, processing time of Exhaustive Method is 9 times larger than the other two methods, memory consumption is 8 times larger. This result shows that a proper treatment of numerical attribute affects processing

Table 2. Difference in Modeling of Numeric Attribute

	CVFDT			$\text{CVFDT}_{\text{NBC}}$		
	Accuracy[%]	Processing Time[s]	Memory [MByte]	Accuracy[%]	Processing Time[s]	Memory [MByte]
Exhaustive	70.86	8.88	339.12			
Gaussian	70.72	1.07	**40.99**	**84.56**	**1.06**	**40.99**
Discretization	70.62	**0.98**	41.68	78.05	1.12	41.71



(a) Accuracy of Each Combination. (b) Processing Time of Each Combination.

Fig. 1. The Effect of Accuracy Based Concept Drift Detection for Several drift levels

time and memory consumption for decision tree induction and that Exhaustive Method is not recommended for practical use in terms of processing time and memory consumption.

In CVFDT, Gaussian Approximation and Discretization Method showed almost the same accuracy. In $\text{CVFDT}_{\text{NBC}}$, however, Gaussian Approximation showed at most 6% higher accuracy than Discretization Method with minor advantages in processing time and memory consumption. Fine division of example space lead to accurate NBCs. Gaussian Approximation calculates finer probability than Discreization Method does, and divides example space finely. This is why Gaussian Approximation shows high accuracy.

Through the experiments, we demonstrated the suitable modeling of numeric attribute for CVFDT and $\text{CVFDT}_{\text{NBC}}$ in terms of accuracy of decision tree, processing time of decision tree induction, and memory consumption. Suitable modeling of numeric attribute for CVFDT is Gaussian Approximation and Discretization method. The one for $\text{CVFDT}_{\text{NBC}}$ is Gaussian Approximation.

(2) The Effect of Accuracy Based Method. To analyze the effect of Accuracy Based Method in terms of accuracy and processing time with different drift levels, we implemented the Accuracy Based Method used in UFFT [3] on CFVDT and $\text{CVFDT}_{\text{NBC}}$ with Gaussian Approximation.

Figure.1(a) and Fig.1(b) shows each method's accuracy and processing time for learning 10,000 examples with various drift level of data stream, where drift level is the percentage of examples that changes their class after concept drift. The higher the drift level, the more examples changes their classes after concept drift. No concept drift takes place when drift level is equal to zero. This paper uses 7% of drift level as a unit, since 7% of drift level is obtained to generate concept drift following Hulten et al [4].

Figure.1(a) shows an interesting fact. In the area where that drift level is between 0% and 7%, Hoeffding Bound Based Method detects concept drift successfully, while the Accuracy Based Method slightly improved accuracy less than 0.5%. In the area where drift level is larger than 14%, however, the Accuracy Based Method overwhelms the Hoeffding Bound Based Method. It is observed in Fig.1(b) that adoption of Accuracy Based Method caused increase of processing time for decision tree induction for CVFDT. It is smaller than 10%, while that in $\text{CVFDT}_{\text{NBC}}$ is larger than 20% reaching 36% in worst case.

Accuracy Based Method should not be applied in the area where drift level is smaller than 7%, because it does not improve the accuracy and degrades the processing time of $\text{CVFDT}_{\text{NBC}}$ down to 36% in the worst case. However, in the area of high drift level, Accuracy Based Method should be applied in spite of the increase in processing time, because the improvement by Accuracy Based Method becomes apparent.

6 Conclusion

In this paper, we compared 3 possible modelings of numeric attribute for CVFDT and $\text{CVFDT}_{\text{NBC}}$, that Exhaustive Method, Gaussian Approximation, and Discretization Mehtod, in terms of accuracy of decision tree, processing time of decision tree induction, and memory consumption. The most suitable treatment of numeric attributes for Hoeffding Tree Algorithm was revealed through experiments. Various concept drift detection methods are proposed, but no performance comparison between them has been reported. This paper evaluated the performance of Accuracy Based Method and Hoeffding Bound Based Method in terms of accuracy and processing time. Through the experiments using artificially generated data, we obtained following results: (1) Gaussian Approximation or Discretization Method should be used on CVFDT, (2) Gaussian Method should be used on HTA with NBC like $\text{CVFDT}_{\text{NBC}}$, (3) Accuracy Based Method should be used under high drift level.

References

1. Domingos, P., Hulten, G.: Mining High-Speed Data Streams. In: Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2000)
2. Gama, J., Rocha, R., Medas, P.: Accurate Decision Trees for Mining High-speed Data Streams. In: Proceedings of the Nineth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 523–528 (2003)
3. Gama, J., Medas, P., Rodrigues, P.: Learning Decision Trees from Dynamic Data Streams. In: Proceedings of the 2005 ACM Symposium on Applied computing, pp. 573–577 (2005)
4. Hulten, G., Spencer, L., Domingos, P.: Mining Time-changing Data Stream. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data mining, pp. 97–106 (2001)
5. Nishimura, S., Terabe, M., Hashimoto, K., Mihara, K.: Learning Higher Accuracy Decision Trees from Concept Drifting Data Streams. In: Proceedings of The Twenty First International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, pp. 179–188 (2008)
6. Pfahringer, B., Holmes, G., Kirkby, R.: Handling Numeric Attributes in Hoeffding Trees. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS, vol. 5012, pp. 296–307. Springer, Heidelberg (2008)
7. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and Unsupervised Discretization of Continuous Features. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 194–202 (1995)

L1 LASSO Modeling and Its Bayesian Inference

Junbin Gao[1,⋆], Michael Antolovich[1], and Paul W. Kwan[2]

[1] School of Computer Science and Accounting,
Charles Sturt University, Bathurst, NSW 2795, Australia
{jbgao,mantolovich}@csu.edu.au
[2] School of Science and Technology,
University of New England, Armidale, NSW 2351, Australia
kwan@turing.une.edu.au

Abstract. A new iterative procedure for solving regression problems with the so-called LASSO penalty [1] is proposed by using generative Bayesian modeling and inference. The algorithm produces the anticipated parsimonious or sparse regression models that generalize well on unseen data. The proposed algorithm is quite robust and there is no need to specify any model hyperparameters. A comparison with state-of-the-art methods for constructing sparse regression models such as the relevance vector machine (RVM) and the local regularization assisted orthogonal least squares regression (LROLS) is given.

1 Introduction

In the setting of supervised learning we are given a set of examples of input vectors $\{\mathbf{x}_i\}_{i=1}^N$ (features) along with the corresponding targets $\{t_i\}_{i=1}^N$. We suppose that input data is of dimension d. Using this data we wish to build a prediction model or learner which will enable us to predict the outcome for new unseen inputs. Of those modeling methods, the least absolute selection and shrinkage operator (LASSO), aiming at providing parsimonious models, has attracted great attention in the past decade. The LASSO method was first introduced by Tibshirani [1]. It has the appealing ability to prune predictors whose effects are actually zero. The LASSO estimate is defined by

$$\widehat{\beta}^{\text{lasso}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{N} (t_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij})^2, \quad \text{Subject to: } \sum_{j=1}^{d} |\beta_j| \leq s. \tag{1}$$

There are many ways to compute the solution of the LASSO such as the Non-Negative Squared algorithm [2] and the scaled Gradient Projection for Sparse Reconstrunction [3]. A recent survey on LASSO is [4]. Cross-validation is a viable tool for estimating the best value of s, but the least angle regression (LAR) procedure [5] turns out to be a better approach. However, although the LAR procedure could identify all the solution paths, we still need to decide the number of predictors to be used in the model.

⋆ The author to whom all the correspondences should be addressed.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 318–324, 2008.

In recent years, the support vector machine (SVM) [6] and kernel machine models (KMM) [7,8] have attracted considerable interest. These techniques have been gaining increasing popularity and are regarded as state-of-the-art techniques. The model used is

$$t = t(\mathbf{x}; \boldsymbol{\beta}) + \epsilon = \beta_0 + \sum_{i=1}^{N} \beta_i k(\mathbf{x}, \mathbf{x}_i) + \epsilon \tag{2}$$

where $k(\mathbf{x}, \mathbf{x}_i)$ is a kernel function and ϵ is an additive noise. The key feature of the SVM is to prune all the unnecessary $k(\mathbf{x}, \mathbf{x}_i)$ and results in a sparse model dependent only on the so-called "support vectors". However, the SVM technique is not always able to construct parsimonious models such as in system identification [9]. This inadequacy motivates researchers to explore new methods for parsimonious models. Tipping [10] first introduced the relevance vector machine (RVM). Chen [8] derived a novel method for constructing sparse kernel models based on the orthogonal least squares (OLS) algorithm [11]. Both are efficient learning procedures for constructing sparse models. There is a lot of literature concerning the problem of regressor selection, see for example [12,13,14,15].

In this paper, we propose an approach by applying the L1 LASSO penalty in (1) to the weights β_j in model (2) to gain model sparsity while adopting a Bayesian learning and inference approach for solving the problem. The rest of the paper is organized as follows. In Section 2, the concepts of L1 LASSO in Bayesian formulation are given and the algorithm for finding L1 LASSO solutions is presented. The experiments are carried out in Section 3, followed by our conclusions in Section 4.

2 Bayesian LASSO Model Formulation

In supervised learning we are given a set of examples of input vectors $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N] \subset \mathbb{R}^d$, d is the dimension number of input space, along with corresponding targets $\mathbf{t} = (t_1, t_2, ..., t_N)^T$ which are independent and identically distributed (iid). We assume that the model noise random variable ϵ follows a Gaussian distribution of mean 0 and variance σ^2. Without loss of generality, we will assume $\beta_0 = 0$ in our discussion.

Let $\mathbf{K}$ be the Gram matrix of the kernel function k defined on the input data X. Due to the assumption of independence of the data points, the likelihood of the complete dataset can be written as

$$p(\mathbf{t}|X, \boldsymbol{\beta}, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{t} - \mathbf{K}\boldsymbol{\beta}\|^2\right\} \tag{3}$$

For the L1 LASSO penalty on weights $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_N)^T$, we impose on $\boldsymbol{\beta}$ a prior of joint Laplacian distribution, defined as

$$p_1(\boldsymbol{\beta}|\lambda) = \frac{(\sqrt{\lambda})^N}{2^N} \exp\left\{-\sqrt{\lambda}(|\beta_1| + |\beta_2| + \cdots + |\beta_N|)\right\}. \tag{4}$$

To develop a generative Bayesian model, we further suppose priors on both the inverse of the noise variance $1/\sigma^2$ and the variance of the Laplacian distribution λ. A suitable prior for the scale parameter $\tau = 1/\sigma^2$ could be a Gamma distribution:

$$p(\tau|a,b) = \text{Gamma}(\tau|a,b) = \frac{b^a}{\Gamma(a)}\tau^{a-1}\exp\{-b\tau\}.$$

In our experiments, we fix its parameters to $a = 500$ and $b = 1$.

Given specifications (3) and (4), the joint distribution of the data set $(X, \mathbf{t})$, the weight variables $\boldsymbol{\beta}$ and the hyperparameter τ (or σ^2) is given by

$$p(\mathbf{t}, \boldsymbol{\beta}, \tau|X) = p(\mathbf{t}|X, \boldsymbol{\beta}, \tau)p_1(\boldsymbol{\beta}|\lambda)p(\tau|a,b) \tag{5}$$

Unfortunately, we cannot compute the posterior of $\boldsymbol{\beta}$ analytically as the denominator $p(\mathbf{t}|\lambda)$ in Bayesian formula necessitates an intractable integration due to the existence of Laplacian distribution. However, the Laplacian distribution $p_1(\beta|\lambda)$ can be expanded as a superposition of an infinite number of Gaussian distributions [16]

$$p_1(\beta|\lambda) = \int_0^\infty \sqrt{\frac{\eta\lambda}{2\pi}}\exp\left\{-\frac{\eta\lambda}{2}\beta^2\right\}p(\eta)d\eta \text{ with } p(\eta) = \frac{1}{2\eta^2}\exp\left\{-\frac{1}{2\eta}\right\} \tag{6}$$

Instead of directly handling the joint distribution defined in (5) we introduce a new latent variable η_i for each β_i as defined in (6) and consider the following joint distribution

$$\begin{aligned} p(\mathbf{t}, \boldsymbol{\beta}, \boldsymbol{\eta}, \tau|X) =& \frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{t} - \mathbf{K}\boldsymbol{\beta}\|^2\right\} \times \frac{b^a}{\Gamma(a)}\tau^{a-1}\exp\{-b\tau\} \\ & \times \prod_{i=1}^{N}\sqrt{\frac{\eta_i\lambda}{2\pi}}\exp\left\{-\frac{\eta_i\lambda}{2}\beta_i^2\right\}\frac{1}{2}\eta_i^{-2}\exp\left\{-\frac{1}{2\eta_i}\right\} \end{aligned} \tag{7}$$

The new distribution can be handled by using a variational Bayesian approach [17,18,19]. We can absorb λ into η_i by setting $\lambda = 1$. The purpose of the variational Bayesian approach is to approximate the posterior $p(\boldsymbol{\beta}, \boldsymbol{\eta}, \tau|\mathbf{t}, X)$ by appropriate independent distributions $Q(\boldsymbol{\beta})Q(\boldsymbol{\eta})Q(\tau)$. $Q(\boldsymbol{\beta})$, $Q(\boldsymbol{\eta})$ and $Q(\tau)$ can be approximated by an iterative procedure defined as follows, $\overline{u}$ means the expectation of u with respect to the approximated posterior $Q(u)$,

1. The best $Q(\boldsymbol{\beta})$ is a Gaussian given by $\mathcal{N}(\boldsymbol{\beta}|\overline{\boldsymbol{\beta}}, \Sigma)$ with the mean $\overline{\boldsymbol{\beta}} = \Sigma^{-1}\overline{\tau}K^T\mathbf{t}$ and covariance $\Sigma = (\text{diag}(\overline{\boldsymbol{\eta}}) + \overline{\tau}K^TK)^{-1}$.
2. The best $Q(\boldsymbol{\eta})$ (or $Q(\eta_i)$) is the Generalized Inverse Gaussian (GIG) distribution given by

$$Q(\eta_i) = G(\eta_i|\lambda, \chi, \psi_i) = \frac{(\psi_i/\chi)^{\frac{\lambda}{2}}}{2K_\lambda(\sqrt{\chi\psi_i})}\eta_i^{\lambda-1}\exp\left\{-\frac{1}{2}(\chi/\eta_i + \psi_i\eta)\right\}$$

where $\lambda = -\frac{1}{2}$, $\chi = 1$ and $\psi_i = \overline{\beta}_i$ and $K_\lambda(\cdot)$ is the modified Bessel function of the second kind. Then $\overline{\eta}_i = 1/\sqrt{\overline{\beta}_i^2}$.

3. The best $Q(\tau)$ is still a Gamma distribution $\text{Gamma}(\tau|a', b')$ where

$$a' = a + \frac{N}{2} \text{ and } b' = b + \frac{1}{2}(\mathbf{t} - K\overline{\boldsymbol{\beta}})^T(\mathbf{t} - K\overline{\boldsymbol{\beta}}) + \frac{1}{2}\text{tr}(K^T K\Sigma)$$

which gives $\overline{\tau} = a'/b'$.

An iterative algorithm can be constructed based on the above three steps which we call the Variational Bayesian Inference Algorithm (VBIA). For those points $(\mathbf{x}_i, t_i)$ whose η_i is increasing in the iterative process, the corresponding weight β_i would approach 0. Unfortunately the convergence speed is quite slow and an improved procedure is desired. We note that in the joint distribution (7) the variable $\boldsymbol{\beta}$ can be analytically integrated out. We can take this advantage to propose a new procedure for the optimal latent variables $\overline{\boldsymbol{\eta}}$ and $\overline{\tau}$.

Define $L = \log p(\mathbf{t}, \boldsymbol{\eta}, \tau|X) = \log \int p(\mathbf{t}, \boldsymbol{\beta}, \boldsymbol{\eta}, \tau|X)d\boldsymbol{\beta}$. Then we find MAP estimate for $\boldsymbol{\eta}$ and τ by maximising L. It is easy to have the exact same iterative formula for $\overline{\tau}$ as in step 3. However we propose the following iterative formula for $\overline{\eta}$

$$\eta_i^{\text{new}} = \frac{1/\eta_i - 3}{\overline{\beta}_i^2 + \Sigma_{ii}} \tag{8}$$

where Σ_{ii} is the ith diagonal element of Σ. That is, instead of using the iterative formula for η_i in step 2, we use (8). We call this algorithm the Modified Variational Bayesian Inference Algorithm (MVBIA) for the LASSO problem.

3 Modeling Examples

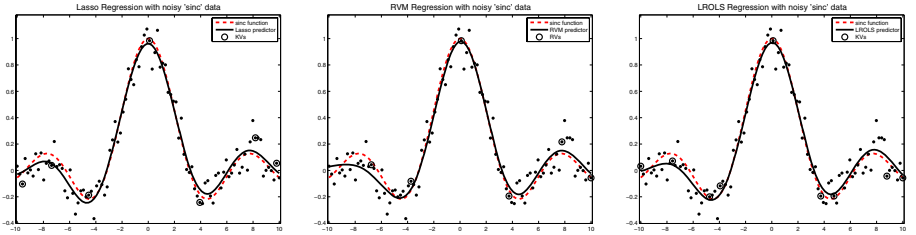
In this section, we will test the new MVBIA method for LASSO regression problems on both a synthetic data set and one real world data set. The experiment will be used to compare MVBIA to two other related algorithms, Chen's LROLS [8] and Tipping's RVM [10].

Example 1: In this example we use a Gaussian radial basis function (RBF) network to model the scalar function

$$\text{sinc}(x) = \sin(x)/x, \qquad -10 \le x \le 10.$$

A set of training data $\{(x_i, t_i)\}_{i=1}^{100}$ is generated for the input x_i by drawing from the uniform distribution over $[-10, 10]$ and target noise within t_i was given by a Gaussian with zero mean and variance 0.1. The target is quite noisy compared to the maximal target values ± 1.

In the experiment, the width of the RBF kernel function is set to 3. The full RBF model (2) is defined by all the RBF regressors with centers at each input training data, thus $N = 100$. The MVBIA method produced a sparser 7-term model. The iterative procedure was terminated at 34 iterative steps when the change of the log value of the successive η_i was less than a given threshold of 0.01.



(a) Model by the MVBIA **(b)** Model by the RVM **(c)** Model by the LROLS

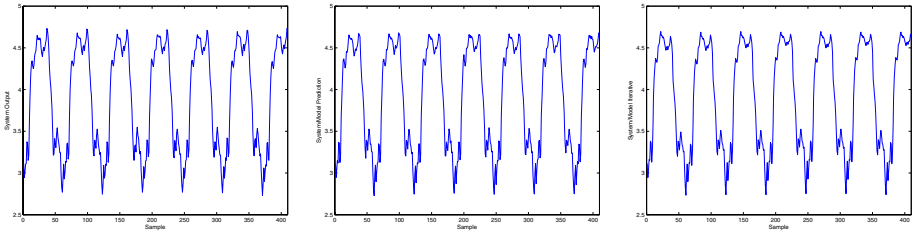
Fig. 1. Dots are the noise training data, the dash-curve is the underlying function $\mathrm{sinc}(x)$, the solid curves are models generated by algorithms and the marker $\circ$ indicates the key regressor vectors selected by each algorithm

Table 1. Mean Square Errors (regrs=regressors)

Example 1			
Methods	LROLS (9 regrs)	RVM (6 regrs)	MVBIA (7 regrs)
Training MSE	0.00137469	0.00180249	0.00126753
Test MSE	0.00137368	0.0245362	0.00126916
Example 2			
Methods	LROLS (34 regrs)	RVM	MVBIA (11 regrs)
Training MSE	0.000435	N/A	0.00045094
Test MSE	0.000487	N/A	0.00047714

Table 1 compares the MSE values over the training and testing sets for the models constructed by the LROLS [8], the RVM [10], and the MVBIA. The result given by MVBIA is comparable to the result generated by both RVM and LROLS algorithms. Although it is reported that the LROLS has less computational cost due to a forward model selection procedure, however in practice we need to run the LROLS at the full size to let each regularizer settle down on the appropriate scale. In practice, after several initial iteration most of the non-significant regressors have been pruned out from the model in MVBIA. The graph of the 7-term model produced by the MVBIA is shown in Figure 1 (a) where the significant vectors (or selected regressors) are marked with $\bigcirc$. From Figure 1 (b) we can see poor performance of the RVM on the left hand side of the curve.

Example 2: In the second example, we are about to construct a model representing the relationship between the fuel rack position (input) and engine speed (output) for a Leyland TL11 turbocharged, direct inject diesel engine operated at low engine speed. A detailed system description and experimental setup can be found in [20]. The data set consists of 410 samples. We use the first 210 data points as training data in modeling and the last 200 points in model validation. An RBF model of the form (2) is used with the input $\mathbf{x}_i$ given by $\mathbf{x}_i = [t_{i-1}, u_{i-1}, u_{i-2}]^T$ where t_{i-1} is the engine speed and u means the fuel input at the last time step.



(a) Engine output data t_i **(b)** One-step prediction **(c)** Iterative prediction

Fig. 2. The results of modelling the relationship between the engine speed and the fuel rack position

The variance of the RBF kernel function was chosen to be 1.69. The total number of regressors is $N = 210$ at the initial stage. The MVBIA algorithm produced a model with 11-term significant regressors. The result is comparable to the one given by LROLS, see Table 1. In our own experiments, a significant effort was made to choose an appropriate value for the parameter ξ used as in LROLS [8]. It is worth pointing out that MVBIA is more robust and insensitive to changes of the initial parameters and is a completely automatic procedure. For example the initial value of η_i is randomly chosen. The results are presented in Table 1.

For this example, the RVM algorithm failed to build up a reasonable model, due to numerical instability of the iterative loop for updating regularization parameters. The constructed RBF model by the MBVIA algorithm was used to generate the one-step prediction t_i of the system output for $i \leq 410$. The iterative model output t_i^d was produced by $\mathbf{x}_i^d = [t_{i-1}^d, u_{i-1}, u_{i-2}]^T$ and $t_i^d = t(\mathbf{x}_i^d; \boldsymbol{\beta})$. The one-step model prediction and iterative model output for this 11-term model selected by the MBVIA algorithm are shown in Figure 2 in comparison with the system output.

4 Conclusions

The MBVIA algorithm has been proposed for solving kernel regression modeling problems with the LASSO penalty. Compared to the LROLS and RVM algorithms the new algorithm is robust with respect to parameter settings. The computational requirements for this iterative model algorithm are very simple and its implementation is straightforward. The core idea can be easily extended to other penalty measures such as robust loss measures and error/loss functions for classification problems.

References

1. Tibshirani, R.: Regression shrinkage and selection via the LASSO. J. Royal. Statist. Soc. B 58, 267–288 (1996)
2. Meinshausen, N.: Relaxed LASSO. Computational Statistics & Data Analysis 52(1), 374–393 (2007)

3. Figueiredo, M., Nowak, R., Wright, S.: Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. IEEE Journal of Selected Topics in Signal Processing 1(4), 586–597 (2007)
4. Hesterberg, T., Choi, N., Meier, L., Fraley, C.: Least angle and L1 regression: A Review. Statistics Surveys 2, 61–93 (2008)
5. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Annals of Statistics 32, 407–451 (2004)
6. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
7. Schölkopf, B., Smola, A.: Learning with Kernels. The MIT Press, Cambridge (2002)
8. Chen, S.: Local regularization assisted orthogonal least squares regression. NeuroComputing 69, 559–585 (2006)
9. Drezet, P., Harrison, R.: Support vector machines for system identification. In: Proceeding of UKACC Int. Conf. Control 1998, Swansea, U.K., pp. 688–692 (1998)
10. Tipping, M.: Sparse Bayesian learning and the relevance vector machine. J. Machine Learning Research 1, 211–244 (2001)
11. Chen, S., Billings, S., Luo, W.: Orthogonal least squares methods and their application to non-linear system identification. International Journal of Control 50(5), 1873–1896 (1989)
12. Kruif, B., Vries, T.: Support-Vector-based least squares for learning non-linear dynamics. In: Proceedings of 41st IEEE Conference on Decision and Control, Las Vegas, USA, pp. 10–13 (2002)
13. Gestel, T., Espinoza, M., Suykens, J., Brasseur, C., deMoor, B.: Bayesian input selection for nonlinear regression with LS-SVMS. In: Proceedings of 13th IFAC Symposium on System Identification, Totterdam, The Netherlands, pp. 27–29 (2003)
14. Valyon, J., Horváth, G.: A generalized LS-SVM. In: Principe, J., Gile, L., Morgan, N., Wilson, E. (eds.) Proceedings of 13th IFAC Symposium on System Identification, Rotterdam, The Netherlands (2003)
15. Suykens, J., van Gestel, T., DeBrabanter, J., DeMoor, B.: Least Square Support Vector Machines. World Scientific, Singapore (2002)
16. Pontil, M., Mukherjee, S., Girosi, F.: On the noise model of support vector machine regression. A.I. Memo 1651, AI Laboratory, MIT (1998)
17. Gao, J., Gunn, S., Kandola, J.: Adapting kernels by variational approach in SVM. In: McKay, B., Slaney, J.K. (eds.) Canadian AI 2002. LNCS (LNAI), vol. 2557, pp. 395–406. Springer, Heidelberg (2002)
18. Gao, J., Xu, R.: Mixture of the robust L1 distributions and its applications. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 26–35. Springer, Heidelberg (2007)
19. Gao, J.: Robust L1 principal component analysis and its Bayesian variational inference. Neural Computation 20, 555–572 (2008)
20. Billings, S., Chen, S., Backhouse, R.: The identification of linear and nonlinear models of a turbocharged automotive diesel engine. Mech. Syst. Signal Processing 3(2), 123–142 (1989)

Discriminating Against New Classes: One-class versus Multi-class Classification

Kathryn Hempstalk and Eibe Frank

Department of Computer Science, University of Waikato, Hamilton, NZ
{kah18,eibe}@cs.waikato.ac.nz

Abstract. Many applications require the ability to identify data that is anomalous with respect to a target group of observations, in the sense of belonging to a new, previously unseen 'attacker' class. One possible approach to this kind of verification problem is one-class classification, learning a description of the target class concerned based solely on data from this class. However, if known non-target classes are available at training time, it is also possible to use standard multi-class or two-class classification, exploiting the negative data to infer a description of the target class. In this paper we assume that this scenario holds and investigate under what conditions multi-class and two-class Naïve Bayes classifiers are preferable to the corresponding one-class model when the aim is to identify examples from a new 'attacker' class. To this end we first identify a way of performing a fair comparison between the techniques concerned and present an adaptation of standard cross-validation. This is one of the main contributions of the paper. Based on the experimental results obtained, we then show under what conditions which group of techniques is likely to be preferable. Our main finding is that multi-class and two-class classification becomes preferable to one-class classification when a sufficiently large number of non-target classes is available.

1 Introduction

Verification problems in machine learning involve identifying a single class label as a 'target' class during the training process, and at prediction time make a judgement as to whether or not an instance is a member of the target class. One-class classifiers seem ideal for this kind of problem; they require nothing other than the target data during training, and make a judgement of *target* or *unknown* for new instances. However, if non-target data is present in the training dataset, it may be beneficial to instead use a multi-class classifier that is able to utilise the negative data in its judgements. A potential disadvantage of this approach in the context of verification is that we are primarily interested in identifying occurrences of completely novel classes at prediction time and multi-class classifiers may not accurately discriminate against these.

In many cases, a one-class classifier is used in preference to a multi-class classifiers simply because it is inappropriate to collect or use non-target data for the given situation. Password hardening—a biometric problem that only has

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 325–336, 2008.

data about one class—is a task where one-class classifiers have been applied to great success [6], and similar research areas including typist recognition [3] and authorship verification [2] have also successfully used one-class classification techniques. One-class classification is often called *outlier detection* (or *novelty detection*) because it attempts to differentiate between data that appears normal and abnormal with respect to the training data. One-class classifiers have also been applied to medical problems, such as tumor detection [4], where a limited quantity of negative data is available during the training process. In contrast, multi-class classifiers have been applied to a huge range of learning problems, including some verification problems where one-class classifiers can also be applied.

With so many techniques available for verification, it is difficult to decide which one will be most effective for discriminating against new 'attacker' classes because it is not obvious how to perform a fair comparison of one-class and multi-class classifiers in this context. We show how it is possible to conduct such a comparison, focusing on a two-class setup as well as a standard multi-class one. Using our method of comparison in conjunction with Naïve Bayes and a corresponding one-class model, we are then able to provide guidelines for choosing a classifier for a given verification problem, where the aim is to discriminate against previously unseen classes of observations.

The next section explores how a fair comparison of the classifiers can be performed. Following this, in Section 3 we describe our experimental setup for testing the different classification techniques, and provide empirical results in Section 4; we also explore situations where one-class classification may be favourable over a multi-class setup, even when negative data is available during the training process. Finally, we conclude the paper in Section 5.

2 Comparing Classifiers Fairly

A standard method for evaluating one-class classifiers is to split a multi-class dataset into a set of smaller one-class datasets, with one dataset per class containing all the instances for the corresponding class. The one-class classifier can then be trained on each dataset in turn, with a small amount of data heldout from the training set and all the other datasets used for testing. Depending on the number of instances available for each class, this generally means that there is a large amount of negative (or 'attacker') data for testing, and a relatively small amount of positive data for both testing and training. In contrast, multi-class classifiers are often evaluated using stratified 10-fold cross-validation, where the data is split into 10 equal-sized subsets, each with the same distribution of classes. The classifier is trained 10 times, using a different fold for testing and the other 9 folds combined for training. These two different evaluation methods are not comparable. In each one the classifier is trained on a different proportion of data for a given class, and is tested on different quantities of data.

In fact, it is not straightforward to perform a fair comparison of one-class classifiers and multi-class ones: one-class classifiers are designed to deal with

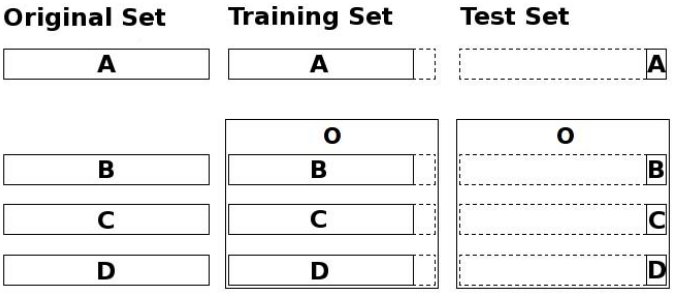


Fig. 1. Standard cross-validation for two-class classification (with relabelling), A is the target class and O is the outlier class

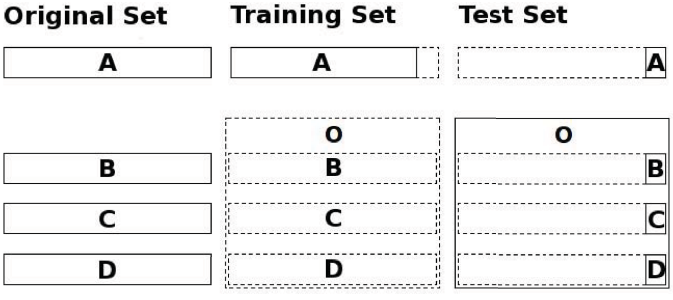


Fig. 2. Standard cross-validation for one-class classification (with relabelling), A is the target class and O is the outlier class

classes that are unseen at training time, but multi-class classifiers typically handle only classes that they have been trained on. A naïve approach to comparing a one-class classifier to a multi-class (more specifically, two-class) classifier[1] is to identify a target class, T, relabel all the data that does not belong to that target class to the label 'outlier', O, and then perform cross-validation on the relabelled dataset—effectively turning a multi-class problem into a two-class one. As shown in Figure 1, this approach is biased in favour of the two-class classifier: a normal cross-validation run will not take into account that O is composed of several classes—meaning that there is a high chance that the test set will contain a class (albeit relabelled) that also occurs in the training set. For one-class classification, we can perform the same cross-validation run, but we delete O from the training set because we only need to train on the target class—as in Figure 2. In all figures missing data is indicated by a dashed line/box.

The practical applications that we are considering in this paper, namely verification problems, have the key feature that entirely new classes are what we want to discriminate against. As we will usually not have training data for these

[1] In this paper we generally refer to multi-class classification whenever there is more than one class involved during the training phase.

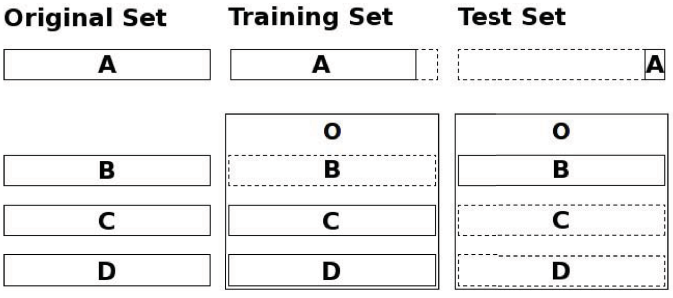


Fig. 3. Cross-validation for unbiased two-class classification (with relabelling), A is the target class, B is the heldout class and O is the outlier class

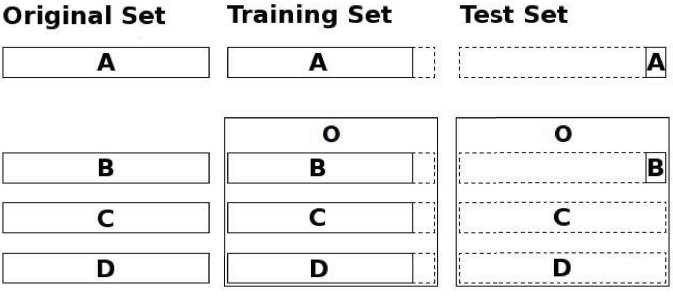


Fig. 4. Cross-validation for biased two-class classification (with relabelling), A is the target class, B is the heldout class and O is the outlier class

new attacker classes, it is inappropriate to use the setup in Figure 1 for evaluating the performance of multi-class techniques: the results will generally be optimistically biased. To remove the bias due to the fact that the multi-class classifier has seen instances of the outlier class previously, the simple answer is to place a heldout class—corresponding to the 'attacker' class—directly into the test set. The training set contains the target class, and all other classes but the heldout class. The test set contains a portion of the target class, and all of the heldout class. All of the non-target classes, including the heldout class, are relabelled to O in both the training and test sets. This is demonstrated in Figure 3. We can use these same datasets for one-class classification, since the one-class classifier does not care about the outlier data in the training set. A drawback of this approach is that we cannot compare the results directly to the biased classifier, since the unbiased classifier is tested with different data. It would be advantageous to be able to perform such a comparison so that the potential benefit of obtaining attacker data for training can be measured—even if this is of mainly academic interest.

Fortunately, there is a way to compare all three types of classification—multi-class biased, multi-class unbiased and one-class—to each other. Let us consider the biased multi-class case first. A target and heldout 'attacker' class are

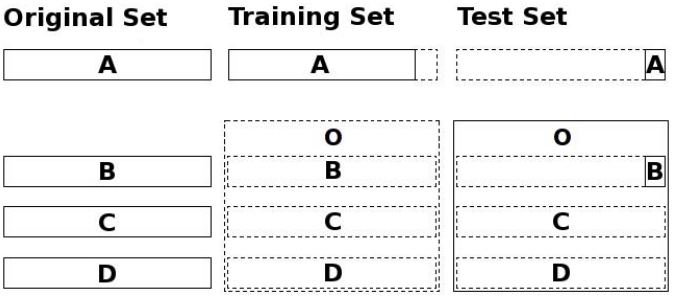


Fig. 5. Cross-validation for one-class classification (with relabelling), A is the target class, B is the heldout class and O is the outlier class

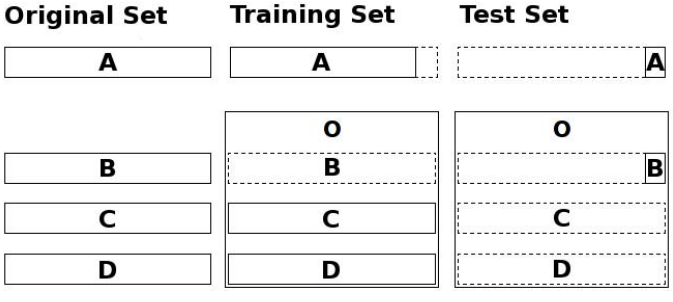


Fig. 6. Cross-validation for unbiased two-class classification (with relabelling), A is the target class, B is the heldout class and O is the outlier class

identified. Then, we perform a normal stratified cross-validation fold, which maintains the class distributions. However, before we relabel the non-target classes, instances from the test set that do not belong to either the target or heldout class are deleted. Finally all non-target classes are relabelled to O, and the evaluation can be performed.

Figure 4 shows the resulting datasets used for multi-class classification. Let us now consider the evaluation of a one-class classifier: it simply ignores all outlier training data, as shown in Figure 5. Lastly, let us consider unbiased multi-class classification. In this case, before the final relabelling is performed the heldout class is removed from the training set, as demonstrated in Figure 6.

The advantage of this approach is that the test set and the target data in the training set is identical for all of the classification techniques, and it is now possible to compare results. Furthermore, as an additional benefit, it is also possible to compare true multi-class classification based on more than two classes with two-class and one-class classifiers by omitting the relabelling step where the non-target classes become class O.

Evaluation of a single target class using different classification techniques (multi-class, two-class and one-class) can be performed by accumulating all predictions for each possible target-heldout class combination. The area under the

ROC curve (AUC) is then calculated for each target class. We use the AUC for comparisons because it is independent of any threshold used by the learning algorithm. To compare classifier performance on an entire multi-class dataset, we use the weighted average AUC, where each target class is weighted according to its prevalence:

$$AUC_{weighted} = \sum_{\forall c_i \in C} AUC(c_i) \times p(c_i) \quad (1)$$

Using a weighted average rather than an unweighted one prevents target classes with smaller instance counts from adversely affecting the results.

3 Evaluation Method

For the experimental comparison, we set up 5 different classification techniques and tested each on UCI datasets [1] with nominal classes. For each dataset 10-fold cross-validation is repeated 10 times. The learning techniques we used are:

1. Biased multi-class classification using Naïve Bayes. No relabelling was performed, data from the heldout class was not removed from the training dataset (as in Figure 4, but without relabelling the non-target classes to O). The test set contained only the target and heldout class.
2. Unbiased multi-class classification using Naïve Bayes. No relabelling was performed, but data from the heldout class was removed from the training dataset (as in Figure 6, but without relabelling the non-target classes to O). The test set contained only the target and heldout class.
3. Biased two-class classification using Naïve Bayes. All non-target classes were relabelled to 'outlier', and the test set contained only the target and (relabelled) heldout class, as in Figure 4.
4. Unbiased two-class classification using Naïve Bayes. All non-target classes were relabelled to 'outlier', the test set contained only the target and (relabelled) heldout class, and instances of the heldout class were removed from the training set, as in Figure 6.
5. One-class classification using a Gaussian density estimate for numeric attributes and a discrete distribution for each nominal one, assuming attribute independence (i.e. 'Naïve Bayes' with only one class). All non-target classes were relabelled to 'outlier' and the test set contained only the target and (relabelled) heldout class, as in Figure 5.

We used Weka's [5] implementation of Naïve Bayes with default parameters for all multi-class and two-class tasks; it is the classifier in Weka that is directly comparable to the one-class classifier we use. The one-class classifier fits a single Gaussian to each numeric attribute and a discrete distribution to each nominal one: a prediction for an instance, X, is made by assuming the attributes are independent. The same happens in Naïve Bayes, but on a per-class basis. In both Naïve Bayes and the one-class classifier missing attribute values are ignored.

4 Results

Using the methodology discussed above, we now present experimental results for UCI datasets. First, we show results obtained by comparing the 5 different classification techniques discussed above. Then we examine some of the results in greater detail, focusing on unbiased two-class classification versus one-class classification.

4.1 Comparison on UCI Datasets

In Table 2 we provide empirical results for all five different classifiers, compared using the weighted average AUC described in Section 2. Only UCI datasets with three or more class labels were used because our evaluation technique requires at least three classes. Table 1 shows some properties of the datasets used in the experiments.

Table 1. Properties of the UCI datasets used in the experiments

	Number of				
			Features		Percentage of
Datasets	Classes	Instances	Nominal	Numeric	Missing Values
anneal	6	898	33	6	-
arrhythmia	16	452	74	206	0.32
audiology	24	226	70	0	2.00
autos	7	205	11	15	1.11
balance-scale	3	625	1	4	-
ecoli	8	336	1	7	-
glass	7	214	1	9	-
hypothyroid	4	3772	23	7	5.36
iris	3	150	1	4	-
letter	26	20000	1	16	-
lymph	4	148	16	3	-
mfeat-factors	10	2000	1	216	-
mfeat-fourier	10	2000	1	76	-
mfeat-karhunen	10	2000	1	64	-
mfeat-morph	10	2000	1	6	-
mfeat-pixel	10	2000	241	0	-
mfeat-zernike	10	2000	1	47	-
optdigits	10	5620	1	64	-
pendigits	10	10992	1	16	-
primary-tumor	22	339	18	0	3.69
segment	7	2310	1	19	-
soybean	19	683	36	0	9.50
splice	3	3190	62	0	-
vehicle	4	846	1	18	-
vowel	11	990	4	10	-
waveform-5000	3	5000	1	40	-
zoo	7	101	17	1	-

Table 2. Weighted AUC results on UCI datasets, for multi-class, two-class and one-class classifiers. Bold font indicates wins for two-class unbiased classification vs. one-class classification and vice versa.

		Classification Techniques				
Datasets	Number of Classes	Multi-class Biased (1)	Multi-class Unbiased (2)	Two-class Biased (3)	Two-class Unbiased (4)	One-class (5)
anneal	6	0.957	0.575	0.948	0.605	**0.788**
arrhythmia	16	0.801	0.724	0.775	**0.723**	0.576
audiology	24	0.960	0.883	0.946	**0.897**	0.881
autos	7	0.831	0.722	0.807	**0.736**	0.567
balance-scale	3	0.970	0.851	0.941	**0.851**	0.806
ecoli	8	0.958	0.855	0.947	0.889	**0.927**
glass	7	0.760	0.680	0.763	0.605	**0.702**
hypothyroid	4	0.931	0.576	0.915	0.587	**0.648**
iris	3	0.994	0.671	0.990	0.671	**0.977**
letter	26	0.957	0.932	0.941	**0.935**	0.887
lymphography	4	0.911	0.432	0.914	0.425	**0.739**
mfeat-factors	10	0.992	0.946	0.975	**0.964**	0.948
mfeat-fourier	10	0.966	0.917	0.949	**0.930**	0.909
mfeat-karhunen	10	0.996	0.969	0.983	**0.976**	0.955
mfeat-morph	10	0.952	0.890	0.948	0.928	**0.941**
mfeat-pixel	10	0.995	0.961	0.981	**0.965**	0.954
mfeat-zernike	10	0.960	0.906	0.946	**0.912**	0.897
optdigits	10	0.986	0.948	0.978	**0.969**	0.959
pendigits	10	0.980	0.915	0.962	0.942	**0.953**
primary-tumor	22	0.839	0.778	0.834	**0.784**	0.732
segment	7	0.971	0.863	0.952	0.863	**0.937**
soybean	19	0.994	0.966	0.988	**0.973**	0.961
splice	3	0.993	0.831	0.983	**0.831**	0.720
vehicle	4	0.767	0.671	0.768	**0.696**	0.658
vowel	11	0.956	0.907	0.926	**0.909**	0.865
waveform	3	0.956	0.692	0.927	0.692	**0.864**
zoo	7	0.999	0.963	0.984	0.963	**0.984**

Table 2 has some noteworthy results, aside from the expected outcome that the biased classification techniques (columns 1 and 3) outperform the unbiased and one-class methods. Of the two biased techniques, one might naïvely expect the two-class approach to perform better: there are less labels and the outlier class contains many of them. However, on all but three of the 27 datasets (glass, lymphography and vehicle), the multi-class classifier either performs the same or better than the two-class classifier—and the difference for these three datasets is not significant. This can be explained by the fact the multi-class Naïve Bayes classifier is able to form a more complex model, with as many mixture components as there are classes. These results suggest that if one does not expect any novel class labels at testing time, one should not merge classes to form a two-class verification problem if Naïve Bayes is used as the classification method.

In situations where the attacker class is not present in the training set (i.e. considering the unbiased classifiers), the picture is different. The multi-class classifier (column 2) scores three wins, six draws and 18 losses against its two-class counterpart (column 4). This result is consistent with intuition: when expecting novel classes during testing, it is safer to compare to a combined outlier class because the multi-class model may overfit the training data. By combining the non-target classes into one class we can provide a more general single boundary against the target class, and increase the chance that a novel class will be classified correctly.

The one-class classifier (column 5) is best compared against the unbiased two-class classifier (column 4) because the latter has been shown to work best in situations where novel classes occur at testing time. As described earlier, one-class classification is intended to deal with novel classes, and learns only the target class during training. One would expect that the two-class classifier could potentially have an advantage because it has seen negative data during training. However, as highlighted in Table 2, the two-class classifier wins on only 16 datasets and the one-class classifier wins on the other 11. On closer inspection, most of the datasets where the two-class classifier wins have a large number of class labels; 12 of those winning datasets have 10 or more original class labels. In contrast, the one-class classifier wins on only two datasets with many class labels—mfeat-morph and pendigits.

4.2 Defining a Domain for One-Class Classification

In order to clarify in which situations one-class classification should be applied, it is instructive to investigate the relationship between the number of class labels available at training time and the accuracy of the two candidate classifiers: the one-class classifier and the unbiased two-class classifier. The number of instances for each class is also relevant; classes with a large number of instances will generally result in a more accurate classifier. However, our primary concern is whether a classifier is capable of identifying novel classes, so it is more appropriate to investigate the effect on accuracy obtained by reducing the dataset size by removing all instances for a particular class label rather than by performing a random selection of instances.

For each of the datasets with 10 or more labels where the unbiased two-class classifier won against the one-class method, we repeated the experimental procedure from Section 3, but on each run we removed a different combination of class labels (and all associated instances) from the training dataset. We gradually increased the number of labels removed, but for each number of labels we considered all possible combinations and calculated the weighted AUC from the accumulated statistics. This was repeated until only two classes remained: the target class, and one original—but relabelled—class. Since we are also removing the heldout attacker class before training, reducing the datasets any further results in no non-target instances present in the training set. The process for producing the test set remains the same—ensuring that the dataset used to obtain the AUC is identical for each method. These results are presented in Table 3;

Table 3. Weighted AUC results obtained by reducing the number of non-target classes in the training set

Dataset	One-class	Original Two-class	Total Removals	Two-class Final AUC	One-class Wins After x Removals
arrhythmia	0.576	0.723	13	0.606	-
audiology	0.881	0.897	21	0.736	7
letter	0.887	0.935	23	0.876	23
mfeat-factors	0.948	0.975	8	0.736	5
mfeat-fourier	0.909	0.949	8	0.726	5
mfeat-karhunen	0.955	0.983	8	0.836	7
mfeat-pixel	0.954	0.965	8	0.876	5
mfeat-zernike	0.897	0.946	8	0.728	4
optdigits	0.959	0.969	7	0.855	4
primary-tumor	0.732	0.784	19	0.740	-
soybean	0.961	0.973	16	0.954	10
vowel	0.865	0.909	8	0.827	8

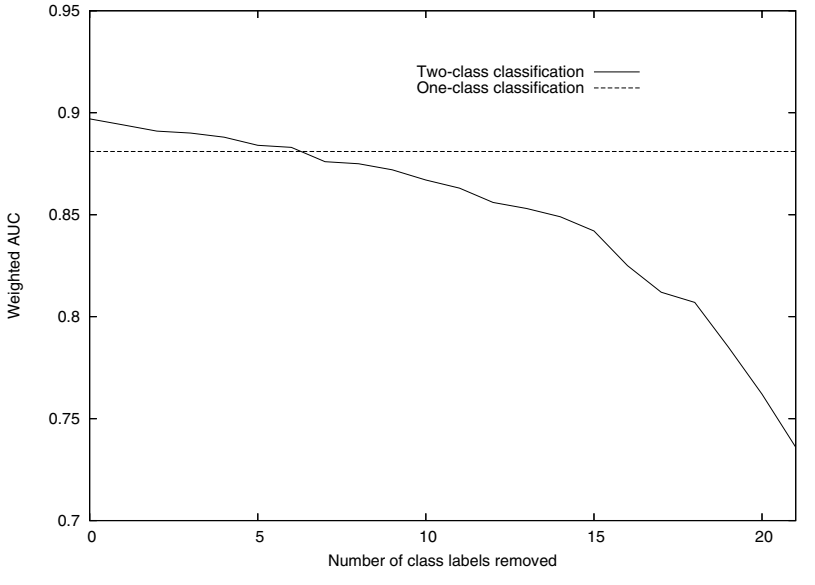


Fig. 7. Number of classes removed versus weighted AUC for the audiology dataset

for brevity we show in the final column the number of classes that were removed before the one-class classifier became better than the two-class one.

For all but two of the datasets shown in Table 3, there exists a point where it is better to use the one-class classifier over the two-class one. This is not unexpected: as the number of non-target classes is reduced, their coverage diminishes until it is no longer worthwhile to use them to define a boundary around the target class. For the two datasets where this is not the case, arrhythmia and

primary-tumor, the density estimate does not appear to form a good model of the data for either classifier and the AUC is relatively low in both cases.

Graphing the results for an individual dataset, such as shown for the audiology dataset in Figure 7, we find that the weighted AUC continually decays as we remove classes from the training data. Of course, the one-class classifier maintains a constant AUC because it does not use non-target data during training. The shape of decay shown in Figure 7 is typical of the datasets in Table 3.

From the results in this section we can say that where there are limited non-target classes available at training time, thus increasing the potential for a novel class to appear that is dissimilar from any existing non-target class, one-class classification should be used in preference to two-class classification. Note that in this situation it may also be beneficial to combine the classifiers in an ensemble—so available outlier training data can be utilised—but we have not investigated this option yet.

5 Conclusions

In this paper we have described how multi-class, two-class and one-class classifiers can be compared to each other by setting up identical test sets and employing the weighted AUC to compare their predictive performance in verification problems with novel classes. We have provided empirical data for each classification technique—Naïve Bayes and the corresponding one-class model—using 27 UCI datasets. Using the results of our experiments, we are able to provide the following advice for use of each classifier in verification problems:

- If no novel classes are expected after the classifier has been trained one should use multi-class classification without relabelling.
- If novel classes are expected, then the training data should be relabelled to 'target' and 'outlier', where the former is the single class we are attempting to verify, and the latter contains all other classes relabelled to a single combined class. If there is a limited number of non-target classes, or they do not sufficiently cover possible novel cases for some other reason, then one should use one-class classification. Otherwise, one should use two-class classification.

In situations where it is unclear whether to use one-class or two-class classification, it may be possible to combine the two classification methods in an ensemble. However, we have not performed any tests regarding this approach, and leave it as a possible avenue for future work. Future work could also include extending empirical results to other learning algorithms. Moreover, we feel it would be interesting to explore further the relationship between the number of non-target instances (rather than the number of classes) and the accuracy of the two-class classifier relative to the one-class classifier.

References

1. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
2. Koppel, M., Schler, J.: Authorship verification as a one-class classification problem. In: Proceedings of the 21st International Conference on Machine Learning, pp. 489–495. ACM Press, New York (2004)
3. Nisenson, M., Yariv, I., El-Yaniv, R., Meir, R.: Towards behaviometric security systems: Learning to identify a typist. In: Proceedings of the 7th International Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 363–374. Springer, Berlin (2003)
4. Tarassenko, L., Hayton, P., Cerneaz, N., Brady, M.: Novelty detection for the identification of masses in mammograms. In: Proceedings of the Fourth International IEEE Conference on Artificial Neural Networks, London, England, pp. 442–447. IEEE, Los Alamitos (1995)
5. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
6. Yu, E., Cho, S.: Novelty Detection Approach for Keystroke Dynamics Identity Verification. In: Liu, J., Cheung, Y.-m., Yin, H. (eds.) IDEAL 2003. LNCS, vol. 2690, pp. 1016–1023. Springer, Heidelberg (2003)

Building a Decision Cluster Classification Model for High Dimensional Data by a Variable Weighting k-Means Method

Yan Li[1], Edward Hung[1], Korris Chung[1], and Joshua Huang[2]

[1] Department of Computing, The Hong Kong Polytechnic University
Hung Hom, Hong Kong, China
{csyanli,csehung,cskchung}@comp.polyu.edu.hk
[2] E-Business Technology Institute, The University of Hong Kong
Pokfulam Road, Hong Kong, China
jhuang@eti.hku.hk

Abstract. In this paper, a new classification method (ADCC) for high dimensional data is proposed. In this method, a decision cluster classification model (DCC) consists of a set of disjoint decision clusters, each labeled with a dominant class that determines the class of new objects falling in the cluster. A cluster tree is first generated from a training data set by recursively calling a variable weighting k-means algorithm. Then, the DCC model is selected from the tree. Anderson-Darling test is used to determine the stopping condition of the tree growing. A series of experiments on both synthetic and real data sets have shown that the new classification method (ADCC) performed better in accuracy and scalability than the existing methods of k-NN, decision tree and SVM. It is particularly suitable for large, high dimensional data with many classes.

Keywords: Clustering, classification, W-k-means, k-NN.

1 Introduction

Classification is a basic task in data mining. As complexity of data increases, the existing techniques for classification face a lot of challenges, for instance, solving the Grand Challenge data mining problems proposed in the recent KDD Panel Report [1]. Therefore, new techniques need to be innovated to deal with large, high dimensional data with multiple classes. Such data occur in many application domains such as text mining, multimedia mining and bio-informatics. This paper proposes an Automatic Decision Cluster Classifier (ADCC) that is designed to achieve that goal.

Clustering methods have been applied to supervised classification problems [2,3]. Several clustering-based classification techniques have been explored. An early example of using the k-means clustering method to build a cluster tree classification model was given in [4]. In this work, a binary cluster tree was built by interactively executing the k-means clustering algorithm. At each node, a further partition was determined by the percentage of the dominant class

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 337–347, 2008.

in the cluster node. However, only small numeric data could be classified and every time only two sub-clusters are formed. In 2000, Huang et al. proposed a new interactive approach to build a decision cluster classification model [5]. In this approach, the k-prototypes clustering algorithm was used to partition the training data, and a visual cluster validation method [6] was adopted to verify the partitioning result at each cluster node. The above two interactive methods are not adequate for high dimensional data with noise because the clustering algorithms used are not able to handle noisy attributes and it is too time consuming to involve human judgment.

In this paper, we propose to use the variable weighting k-means (W-k-means) algorithm [7] to build cluster-based classification models automatically. Because W-k-means is able to reduce the impact of noisy attributes by assigning smaller weights to them in clustering, it implicitly performs attribute selection in the clustering process. Meanwhile, the weight information can also be used in classification. As such, W-k-means is more suitable for high dimensional data with noisy attributes. Another improvement from the previous methods is that in the tree growing process we use Anderson-Darling test to determine whether a node can be further partitioned or not. In this way, distribution of the training samples at each node is considered together with the percentage of the dominant class used in the previous methods. Anderson-Darling test replaces the visual cluster validation method as in [5] so the tree building process is automated.

A series of experiments on both synthetic and real data sets were conducted to demonstrate the efficiency and accuracy of the ADCC method. Compared with other classification methods, including k-NN, J48 (a decision tree algorithm) and SMO (one of SVM algorithms), our experimental results have shown that the ADCC method has performed better than other methods in both classification accuracy and scalability on large high dimensional sparse data sets. Thus the results demonstrate that the ADCC method is more suitable for large, high dimensional data with many classes.

The rest of this paper is organized as follows. In Section 2, we introduce the decision cluster classification model and an algorithm to build such model. In Section 3, experimental results and comparisons are reported. In Section 4, we conclude this paper.

2 An Automatic Decision Cluster Classification Method

In this section, we present the DCC model and the ADCC method for building a cluster tree, selecting a model and using the model for classification.

2.1 Definitions

The following is the proposition of the DCC model. Given a training data set, objects in the same class tend to be spatially close in the data space. Objects in the same cluster generated from the training data by a clustering algorithm have similar behaviors or properties and tend to be in the same class [5].

Let $X = x_1, x_2, ..., x_n$ be a training data set of n classified objects, each described by m attributes and labeled by one of c classes. The following definitions were modified from [5].

Definition 1. A *k-clustering* of X is a partition of X into k clusters C_1, C_2,..., C_k, which satisfies: $C_i \neq \emptyset$, $i = 1, ..., k$, $\bigcup_{i=1}^{c} C_i = X$, and $C_i \bigcap C_j = \emptyset$, $i \neq j$, $i, j = 1, .., k$.

Definition 2. The dominant class in a cluster is the class that the majority of objects are labeled to. A cluster with a dominant class is called a decision cluster. The percentage of the dominant class in the cluster defines the confidence level of the decision cluster.

Definition 3. A decision cluster classification (DCC) model consists of (*i*) a subset of decision clusters generalized from the whole training data set, and (*ii*) a defined distance function.

The distance function is used to compute the distances between an object to be classified and the centers of decision clusters.

In principle, any subset of decision clusters can form a DCC model. However, the model performance on classification accuracy depends on the decision clusters generated by the clustering process and selection of the subset. Therefore, the following two processes are crucial: (1) generation of a set of decision clusters and (2) selection of a subset for the model. Below, we present a method to generate a set of nested clusters that form a cluster tree for model selection.

2.2 Decision Cluster Tree

For large complex data, a large number of decision clusters need to be generated from the training data. Considering the varying densities in data distribution, we propose a recursive process to generate a sequence of nested clusterings to build a cluster tree for model selection.

Definition 4. A cluster tree is a sequence of clusterings, and for any two levels p, q with $p < q$ (i.e., level p is higher than level q) and for any cluster C_j^{q-1} in level q, there is a cluster C_i^{p-1} in level p such that $C_j^{q-1} \subset C_i^{p-1}$. We say that the clustering of level q is nested in the clustering of level p.

Figure 1 shows an example of a cluster tree of four levels. Level 0 is the root T which is the training data set. The root T is partitioned into 3 clusters C_1^0, C_2^0, C_3^0. Here, the superscript indicates the level of the node from which the clusters are generated and the subscript is the cluster number in this level. Clusters C_1^0 and C_3^0 are further partitioned into 2 and 3 sub-clusters respectively, which form level 2 of the cluster tree. Subsequently, two clusters C_2^1 and C_4^1 are further partitioned into three and two sub-clusters respectively, which form level 3 of the cluster tree. This tree can be represented as the following sequence of nested clusterings

$$T(C_1^0(C_1^1, C_2^1(C_1^2, C_2^2, C_3^2)), C_2^0, C_3^0(C_3^1, C_4^1(C_4^2, C_5^2), C_5^1))$$

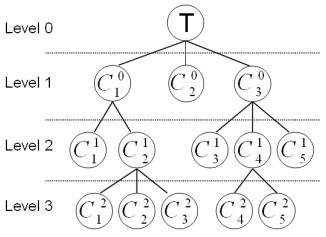


Fig. 1. Example of a cluster tree

Given a training data T and a clustering algorithm **f**, a cluster tree can be generated by recursively calling **f** to partition the nodes, starting from the root node. To enable this process automatically, we have to resolve the following issues: (1) Which clustering algorithm to generate a clustering at each node? (2) How many clusters to generate at each node? (3) Where to stop at each path of the tree? In the following, we will discuss the techniques used to solve these problems and the ADCC algorithm that implements these techniques to automatically build a cluster tree from a training data set.

The W-k-means [7] algorithm was adopted to build a cluster tree because it is efficient and able to automatically compute the attribute weights from the training data to reduce the effect of noisy attributes. The number of sub-clusters and the initial centers must be specified before executing W-k-means. The ADCC algorithm uses the function $K\text{-}Selection(X, \alpha)$ to return the number of sub-clusters for current cluster, where X is the current cluster and α is the threshold (we use 0.1%). $K\text{-}Selection(X, \alpha)$ computes the percentage of each class and returns k as the number of classes with a percentage greater than α. Then, for each of the k classes, we compute the centers of its samples as its initial center. This is called **supervised selection** which, instead of random selection, is implemented in the ADCC algorithm as function $C\text{-}Selection(k, X)$, where k is the number of sub-clusters generated by $K\text{-}Selection(X, \alpha)$.

We use multiple termination conditions: the cluster size, class purity and data distribution to determine whether a node will be further partitioned or not. We assume that a good cluster has a normal distribution, so we adopted Anderson-Darling test [8] to test the data distribution. The algorithm $Terminal\text{-}Test(X)$ for stopping tree growing is given in Table 1.

Table 2 shows the ADCC algorithm to automatically build decision cluster classification models with the above techniques.

2.3 Model Selection and Classification

After a decision cluster tree is built, any subset of disjoint decision clusters makes a DCC model. There are many ways to select a classification model from a decision cluster tree. In this work, we select the leaf nodes of the decision cluster tree.

Table 1. Algorithm of $Terminal\text{-}Test(X)$

Input: the node X which contains n objects.
Output: a Boolean value $Termi$ which is either
(i) TRUE which means "stop" or (ii) FALSE, otherwise.
Remarks:
δ: the threshold of the number of data in X (e.g., 10).
β: the threshold of the frequency of a class in X (e.g., 90%).
λ: the threshold for AD-Test on X.
Begin
1. $Termi$ = FALSE;
2. if ($n < \delta$ OR the frequency of the dominant class $> \beta$ OR $AD\text{-}Test(X) < \lambda$)
3. $Termi$ = TRUE and label X with the dominant class label;
4. return $Termi$;
End

The classification model is used to classify new objects as the following: (1) Define a distance function specific for classification; (2) Compute the distances between a new object and the centers of the decision clusters in the model; (3) Identify the decision cluster with the shortest distance to the object. Assign the label of the decision cluster to the new object as its class.

In this work, we use the weighted Euclidean distance function as follows

$$d = \sqrt{\sum_{i=1}^{n} (w_i(x_i - y_i))^2} \tag{1}$$

where w_i is the weight for the $i\text{-}th$ attribute. The weights are computed when we cluster the training data by the $W\text{-}k$-means clustering algorithm [7]. Since the weight distribution is different when we cluster a different node, here we adopt the weight distribution computed when we cluster the root of the decision cluster tree.

3 Experiments

We implemented the ADCC algorithm in java and conducted a series of experiments on both synthetic and real data sets. We compared the results obtained by the proposed method with those from other classification methods, including $k\text{-}NN$, decision tree and SVM. Weka[1] implementations of these algorithms were used in our comparisons. All experiments were conducted on an Intel Core 2 Duo CPU, 2.33 GHz computer with a 2GB memory.

[1] http://www.cs.waikato.ac.nz/∽ml/weka/

Table 2. ADCC Algorithm

Input: A training data set T (with m dimensions and c classes).

Output: A classification model *ADCC-model*.

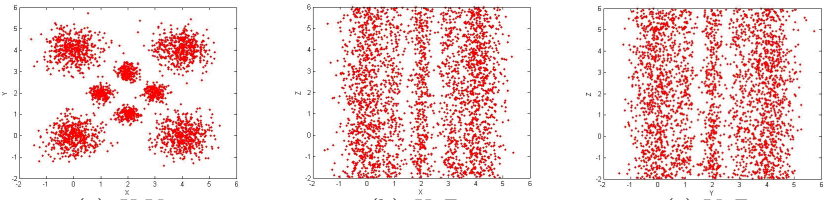
1. initialize a decision cluster tree DCT with root {T};;
2. sign the root as *internal node*;
3. for each internal node X in DCT
4. if ($Terminal\text{-}Test(X)$) sign X as *leaf node*;
5. $k = K\text{-}Selection(X, \alpha)$;
6. $CENTER\text{-}ARRAY = C\text{-}Selection(k, X)$; //Compute initial centers
7. run W-k-means on X with k and $CENTER\text{-}ARRAY$;
8. sign k sub-clusters as *internal node*;
9. add k sub-clusters into DCT;
10. end for
11. extract all leaf nodes from DCT as classification model and represent each node by its center;

3.1 Experiments on Synthetic Data

The purpose of our experiments on synthetic data is to demonstrate and compare the classification performance and efficiency of using two different initial cluster center selection methods and two different clustering algorithms (the standard k-means and W-k-means). The results show that our method is efficient and can reduce the impact of noisy attributes. Then we conducted experiments to test the scalability of ADCC. The detailed results are described below.

The first synthetic data contains 3400 objects in a 3-dimensional space, and is categorized in 8 classes. The first two dimensions X, Y contain 8 normally distributed clusters and the points in each cluster were categorized as the same class. The third dimension Z is a noisy dimension in which all points are uniformly distributed. Each of four clusters has 500 objects and each of other four clusters has 200 objects. 600 noisy points are randomly generated. Figure 2 shows the data set projected in different combinations of two-dimensions. We can observe the impact of the noisy dimension Z in Figure 2(b) and (c).

In this synthetic data set, we first tested two initial cluster center selection methods: random selection and supervised selection. Table 3 shows the results measured with Recall, Precision and F-Measure. We can see that except for the precision of class C_2, all results produced with the supervised selection were better than the random selection. Secondly, we compared the standard k-means and W-k-means algorithms on this synthetic data set. The classification model from the standard k-means algorithm used Euclidean distance to classify new objects whereas the model from the W-k-means algorithm used the weighted Euclidean distance. The comparison results are shown in Table 4. We can see



(a) X,Y (b) X,Z (c) Y,Z

Fig. 2. Projection of the data set on different subspaces

Table 3. Classification results from random selection and supervised selection of initial cluster centers (R for random selection, P for supervised selection)

		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
Recall	R	0.907	0.71	0.831	0.719	0.714	0.877	0.818	0.883
	P	0.941	1	0.926	1	1	1	1	0.982
Precision	R	0.77	0.875	0.817	0.908	0.733	0.841	0.875	0.883
	P	1	0.786	1	0.941	0.875	1	0.985	1
F-Measure	R	0.833	0.784	0.824	0.802	0.724	0.859	0.846	0.883
	P	0.97	0.88	0.962	0.97	0.933	1	0.979	0.991

Table 4. Impact of variable weighting on classification accuracy

		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
Recall	k-means	0.938	0.53	0.909	0.65	0.744	0.987	0.762	0.823
	W-k-means	0.968	0.579	0.968	0.69	0.733	0.989	1	0.957
Precision	k-means	0.997	0.469	0.998	0.577	0.512	0.993	0.826	0.923
	W-k-means	1	0.527	1	0.663	0.753	1	0.824	1
F-Measure	k-means	0.967	0.498	0.952	0.611	0.607	0.99	0.793	0.87
	W-k-means	0.984	0.552	0.984	0.667	0.743	0.955	0.903	0.978

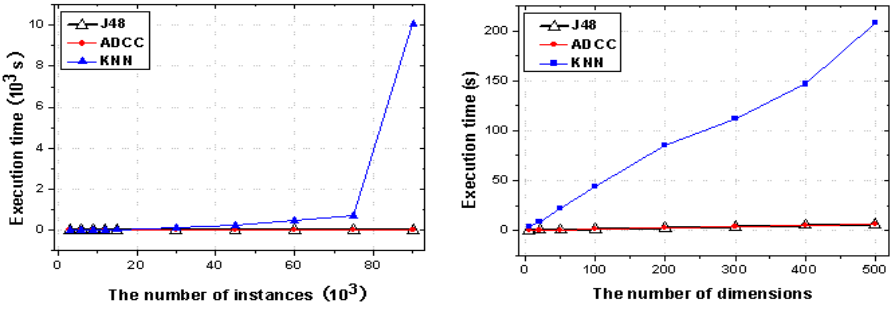
that W-k-means with variable weighting performed better in most classes. With cross validation, the total average accuracy was 85.46% for k-means and 90.82% for W-k-means.

These results have shown that supervised selection for initial centers and W-k-means were indeed more effective than random selection and k-means in producing better classification models.

To test the scalability of ADCC, we generated two groups of synthetic data sets with different numbers of dimensions and instances (shown in Table 5). Each data set contains three clusters randomly generated with normal distributions. In each run, we used 70% of data as training data and the remaining 30% as

Table 5. Two groups of synthetic data sets (each having three classes)

Data sets	Dimensions	Instances	Data sets	Dimensions	Instances
A1	5	5,000	B1	4	3,000
A2	20	5,000	B2	4	9,000
A3	50	5,000	B3	4	15,000
A4	100	5,000	B4	4	30,000
A5	200	5,000	B5	4	45,000
A6	300	5,000	B6	4	60,000
A7	400	5,000	B7	4	75,000
A8	500	5,000	B8	4	90,000



(a) Execution time vs data size (b) Execution time vs dimension number

Fig. 3. Scalability comparison between ADCC and *k-NN*

test data. For each data set, we recorded the total execution time. We also ran the basic *k-NN* algorithm on these data sets.

Figure 3(a) shows the execution time against the number of instances and Figure 3(b) shows the execution time against the number of dimensions. We can see that the execution time increased linearly for ADCC whereas the execution time for *k-NN* increased rapidly when the number of instances approached 75000. Although the execution time for *k-NN* increased linearly as the number of dimensions increased, the increase in speed was much faster than ADCC. Meanwhile, the execution time for ADCC and decision tree (J48) are comparable. Therefore, ADCC is scalable on large data set.

3.2 Experiment on Real Data

We compared ADCC with other well-known classification algorithms on five real data sets. *Waveform*, *Reuters*, *Sub1* and *Sub2* were taken from the UCI machine learning data repository [2]. For the *Reuters* data set, the standard document

[2] http://archive.ics.uci.edu/ml/datasets.html

Table 6. Real data sets

Data set name	Instances	Dimensions	Classes	Training	testing
Waveform	5000	40	3	3500	1500
Reuters	9980	337	10	6986	2994
Leukemia3	327	12558	7	215	112
Sub1	400	7904	4	280	120
Sub2	299	6570	4	209	90

frequency method was used to select relevant attributes from the original feature space. *Sub1* and *Sub2* were built from the 20-Newsgroup data, and the words in each document were weighted by the standard $tf \cdot idf$. *Leukemia3*, which is related to studies of human cancer, was taken from K-TSP Program Download Page [3]. K-TSP Program includes 10 multi-class gene expression data sets. For our decision cluster tree model to be more effective, the training samples should not be too few so that there can be more than a few members in each cluster. Thus, we select *Leukemia3* which has more training samples than other data sets in K-TSP Program. Table 6 lists the characteristics of these real data sets.

In this experiment, we compare the performance of ADCC with three well-known classification methods: *k-NN*, decision tree (J48) and SVM (SMO). For *k-NN*, the number of neighbors, k, is equals to 1. We use default parameters for J48 and SMO in Weka. For J48, the minimum number of instances per leaf is set as 2. The SMOs are trained with a poly kernel where the complexity parameter C is set as 1.0. For the *Waveform*, *Reuters*, *Sub1* and *Sub2*, we used 70% of data as training data and 30% as test data. We used the ADCC algorithm to build the DCC models. When we ran ADCC on *Waveform* and *Reuters*, we set the termination test thresholds as follows $\delta = 10$ (the threshold of the number of data in a node); $\beta = 90\%$ (the threshold of the frequency of a class in a node); $\lambda = 500$ (the threshold for AD-Test). For *Sub1* and *Sub2*, we set $\delta = 5$; $\beta = 90\%$; $\lambda = 5$. These training data sets are smaller than the first two data sets, so we choose a smaller δ. In the experiment on *Leukemia3*, we use the training data set and test data set originally contained in K-TSP Program. We set the termination test thresholds as follows $\delta = 2$; $\beta = 90\%$; $\lambda = 5$. From our experiments, the value of parameter λ is not very critical for the results.

The execution time (in seconds) and classification accuracy generated by the four classification methods from these real data sets are listed in Table 7. We recorded the total execution time for ADCC and *k-NN*, and only the training time for decision tree and SVM were recorded because Weka only provides the training time. From this table, we can see that for data set *Waveform*, the accuracy of ADCC was higher than those of *k-NN* and J48 (decision tree method) and lower than SMO (SVM). ADCC was much faster than *k-NN* but slower than the other two. Comparatively, this data set was simpler with fewer

[3] http://jshare.johnshopkins.edu

Table 7. Classification results of two real world data sets by four classification methods

Data sets	Waveform		Reuters		Leukemia3		Sub1		Sub2	
Metrics	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.
ADCC	2.516	0.838	74.952	0.6897	16.631	0.9196	13.25	0.775	6	0.7111
k-NN	15.688	0.708	485.625	0.5795	21.094	0.7589	20.109	0.4833	11.219	0.3444
Decision tree	1.2	0.7326	35.23	0.6529	9.53	0.7589	17.75	0.575	9.63	0.6777
SMO	1.94	0.85	392.91	0.6586	24.38	0.8392	35.63	0.7	19.61	0.6555

dimensions and instances and a small number of classes. However, for data set *Reuters* which was more complex with more instances and classes, and much higher dimensions, ADCC outperformed all other algorithms in classification accuracy. It is much faster than k-NN and SVM, but only slightly slower than the decision tree implementation. For the data set *Leukemia3*, *Sub1* and *Sub2* which have very high dimensions, ADCC outperformed other algorithms obviously. Therefore, ADCC is more suitable for large, high dimensional data with many classes and can be used in text mining and bioinformatics.

4 Conclusion

In this paper, we have proposed a new classification framework for using a clustering method to build decision cluster classification models. We have presented an automatic algorithm ADCC which uses the variable weighting k-means algorithm W-k-means to build a cluster tree from a training data set. ADCC is an enhancement to the interactive approach presented in [5]. In this automatic approach, we have proposed solutions to solve three important problems: (1) selection of the number of clusters at each node, (2) selection of the initial cluster centers, and (3) termination of further clustering at a leaf node.

We have presented experimental results on both synthetic and real world data sets and compared the performance of ADCC with those of other well-known classification methods. The comparison results have shown that ADCC has advantages in classifying large, high dimensional data with multiple classes.

References

1. Piatetsky-Shapiro, G., Djeraba, C., Getoor, L., Grossman, R., Feldman, R., Zaki, M.: What are the grand challenges for data mining? KDD-2006 panel report. SIGKDD Explorations 8, 70–77 (2006)
2. Kyriakopoulou, A., Kalamboukis, T.: Text classification using clustering. In: ECML-PKDD Discovery Challenge Workshop Proceedings (2006)
3. Zhang, B., Srihari, S.N.: Fast k-nearest neighbor classification using cluster-based trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 525–528 (2004)
4. Mui, J., Fu, K.: Automated classification of nucleated blood cells using a binary tree classifier. IEEE Transactions on Pattern Analysis and Machine Intelligence 2, 429–443 (1980)

5. Huang, Z., Ng, M., Lin, T., Cheung, D.: An interactive approach to building classification models by clustering and cluster validation. In: Leung, K.-S., Chan, L., Meng, H. (eds.) IDEAL 2000. LNCS, vol. 1983, pp. 23–28. Springer, Heidelberg (2000)
6. Huang, Z., Lin, T.: A visual method of cluster validation with fastmap. In: Terano, T., Chen, A.L.P. (eds.) PAKDD 2000. LNCS, vol. 1805, pp. 153–164. Springer, Heidelberg (2000)
7. Huang, J., Ng, M., Rong, H., Li, Z.: Automated variable weighting in k-means type clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 657–668 (2005)
8. Anderson, T.W., Darling, D.A.: Asymptotic theory of certain "goodness-of-fit" criteria based on stochastic processes. The Annals of Mathematical Statistics 23, 193–212 (1952)

Locality Spectral Clustering

Yun-Chao Gong[1] and Chuanliang Chen[2]

[1] Software Institute, Nanjing University
[2] Department of Computer Science, Beijing Normal University
gyc05@software.nju.edu.cn, c.l.chen86@gmail.com

Abstract. In this paper, we propose a novel spectral clustering algorithm called: Locality Spectral Clustering (LSC) which assumes that each data point can be linearly reconstructed from its local neighborhoods. The LSC algorithm firstly try to learn a smooth enough manifold structure on the data manifold and then computes the eigenvectors on the smooth manifold structure, then as former spectral clustering methods, we use the eigenvectors to help the k-means algorithm to do clustering. Experiments have been performed on toy data sets and real world data sets and have shown that our algorithm can effectively discover the cluster structure and holds much better clustering accuracy than former methods. It is also worth noting that our algorithm is also much more stable in parameter than former spectral clustering methods.

1 Introduction

In recent years, clustering methods are becoming more and more important. A new kind of clustering methods called the spectral clustering methods have been proposed recently. The spectral clustering methods have one advantage that it do not require to estimate an explicit model of data distribution [6]. It uses a spectral analysis of the matrix of the data instances. The spectral methods was first proposed in [2,3,4,5].

Although the spectral clustering methods have many advantages, they still have many drawbacks to be addressed. As pointed in [6], one problem in spectral clustering methods is that it is very difficult to estimate the parameter σ in the algorithm [6]. In many spectral clustering methods [2,3,4], they all firstly define a k-nearest neighbor graph W which the edge weight e_{ij} (*i.e.* the edge links data x_i and x_j) of W is calculate by the Heat kernel: $e_{ij} = exp(-\|x_i - x_j\|^2/(2\sigma^2))$. However in the Heat kernel, the bandwidth σ of the *Gaussian Function* will affect the clustering results significantly. It is very difficult to estimate a good parameter σ for clustering tasks when there are no labeled examples.

Many former works have been proposed to deal with the above problem. [4] proposed a method which tries to choose a exact σ by running the clustering algorithm for a number of times and select the best parameter which provides the least distorted clusters of the normalized eigenvectors, however the method significantly increases the computational time. [6] proposed a simple method which use a *local scale* to compute the affinity between each pair of points. All

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 348–354, 2008.

these above methods only focus on how to choose a good σ to the spectral clustering algorithms but in this paper, we present a novel method which does not need the parameter σ.

In this paper we propose a novel spectral clustering algorithm called **Locality Spectral Clustering** (LSC). The idea of LSC is that we assume in real worlds data set, the nearby points are likely to belong to the same class and the data points on the same data structure (*i.e.* a cluster or a manifold) may belong to the same class [8,15]. So that in this paper, we can apply a manifold based approach to construct the graph for spectral clustering tasks [10,13]. We firstly try to learn a smooth enough data manifold on the original data and then analyze the eigenvectors on the smooth manifold structure. Then the eigenvectors can be used to help the classical k-means algorithm to do clustering. The method used to learn the smooth graph W in this paper was firstly proposed in [7] which use the graph to do semi-supervised classification.

The rest of this paper is organized as follows: Section 2 will described our Locality Spectral Clustering (LSC) algorithm in detail. The experimental results and discussion of the toy data sets and real world data sets are shown in Section 3. Conclusions are made in Section 4.

2 The Proposed Algorithm

The LSC algorithm can be separated into three steps: (1) Construct a smooth enough graph W for spectral clustering, (2) solve the eigenvectors from the graph W, (3) use the eigenvectors to help k-means to clustering. We will first introduce how to construct the graph W and then describe the rest steps.

2.1 Locality Graph for Spectral Clustering

The graph was firstly proposed in [7] for semi-supervised classification. We first introduce some notations: Given a set of data samples $X = \{x_1, \ldots, x_n\}$ in R^l represents a set of n data objects. c is the number of clusters we want to cluster. Construct the neighborhood graph $G =< V, E >$, where each vertex in the graph denotes a data instance x_i and E is the edge set. The edge e_{ij} denotes the relationship between two data instances x_i and x_j. The e_{ij} used in many former methods [1,2] is always calculated by:

$$e_{ij} = e^{-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}} \tag{1}$$

However in [15], it indicates that there is no reliable way to choose the parameter σ when there are very few or even there are no labeled examples. Also, as have shown in our experiments, the parameter σ is very unstable and a small change will greatly affect the clustering result. Thus we apply a more reliable and stable way to construct the graph W for spectral clustering. We propose to use the local linear neighborhood information of each point to construct W. Hence the objective is to minimize:

$$J = \sum_i \left\| x_i - \sum_{j:x_j \in \mathcal{N}(x_i)} w_{ij} x_j \right\|^2 \tag{2}$$

where $\mathcal{N}(x_i)$ denotes all the neighborhoods of x_i, and w_{ij} is the contribution of x_j to x_i. In [7] it constrain $\sum_j w_{ij} = 1, w_{ij} \geq 0$ to guarantee the weights are nonnegative. Thus the w_{ij} are used to measure how similar x_j to x_i. Then it can be easily inferred that:

$$\begin{aligned} J_i &= \left\| x_i - \sum_{j:x_j \in \mathcal{N}(x_i)} w_{ij} x_j \right\|^2 = \left\| \sum_{j:x_j \in \mathcal{N}(x_i)} w_{ij}(x_i - x_j) \right\|^2 \\ &= \sum_{j,k:x_j,x_k \in \mathcal{N}(x_i)} w_{ij} w_{ik} (x_i - x_j)^T (x_i - x_j) \\ &= \sum_{j,k:x_j,x_k \in \mathcal{N}(x_i)} w_{ij} G^i_{jk} w_{ik} \end{aligned} \tag{3}$$

where G^i_{jk} denotes the (j,k)-th entry of the local Gram matrix of point x_i: $(G^i)_{j,k} = (x_i - x_j)^T (x_i - x_j)$. Thus weights of the above objective function can be solved by the following n standard Quadratic Programming (QP) problem:

$$\begin{aligned} \min_{w_{ij}} & \sum_{j,k:x_j,x_k \in \mathcal{N}(x_i)} w_{ij} G^i_{jk} w_{ik} \\ s.t. & \sum_j w_{ij} = 1, w_{ij} \geq 0 \end{aligned} \tag{4}$$

Then we can construct the graph for spectral clustering as follows: for each data point, we set the weights of its neighborhoods as: $W_{ij} = w_{ij}$ and $W_{ji} = w_{ij}$. Then if there are any overlaps, the later weight will overlap the former one thus we can guarantee the symmetric of the matrix. Then the W can be used as the weight matrix of G.

2.2 The Locality Spectral Clustering (LSC) Algorithm

Then we can use the classical spectral clustering procedure [4] to do clustering with the graph W constructed in the first step. With the graph W we can solve the generalized eigenvectors $u_1, \ldots, u_c$ from the generalized eigen-problem $Lu = \lambda Du$ with the *Normalized Laplacian* $L = D^{-1/2} W D^{-1/2}$. Then we re-normalize the rows of the eigenvectors and each row of the normalize eigenvectors are used as a data point in $\mathcal{R}^c$. Then we use the k-means method to cluster the points $(x_i)_{i=1,\ldots,n} \in X$ into c clusters. The main LSC procedure have be summarized in Table 1.

The main advantage of our LSC algorithm is that by taking the advantage of the smooth graph W, LSC can effectively and correctly discover the underlying cluster structure. Furthermore the parameter k in the LSC algorithm is very stable and the algorithm always holds much better accuracy than former methods.

Table 1. The LSC Algorithm

Input:
Given a set of data points $X = \{x_1, \ldots, x_n\}$ in R^l

Parameter:
k: The number of nearest neighbors
c: The number of clusters want to cluster

Procedure:
1. Construct the graph W according to Eq. 5 by solving Eq.4.
2. D is defined to be a diagonal matrix $D_{ii} = \sum_{j=1}^{n} W_{ij}$ and construct the *Normalized Laplacian* $L = D^{-1/2} W D^{-1/2}$.
3. Solve the first c generalized eigenvectors $u_1, \ldots, u_c$ of the eigen-problem: $Lu = \lambda Du$ and form a matrix $U = [u_1, \ldots, u_c] \in \mathcal{R}^{n \times c}$
4. Re-normalize the rows of U that $T_{ij} = U_{ij} / (\sum_j U_{ij}^2)^{1/2}$ to get the unit length yielding T.
5. Each row of T is viewed as a data point and cluster the points $(x_i)_{i=1,\ldots,n}$ with the k-means algorithm into c clusters.

3 Experiments

In this section, we will report our results on some toy data sets and the real worlds image data sets (COREL) and then give some discussion. We first report the results on the toy data sets and then we will report the results on the COREL data set. In the experiments we use the *Normalized Mutual Information* (*NMI*) [12] to evaluate the results of clustering.

3.1 Results on Toy Data Set

In this experiment, we tested our algorithm with some typical toy data sets. The results are summarized in Fig. 1. The odd rows are results by Ng-Jordan-Weiss (NJW) Algorithm [4] and the even rows are results by our LSC algorithm. We increase the parameter σ of the NJW algorithm from 0.1 to 10 respectively and increase the parameter k of our LSC algorithm from 5 to 25 respectively. It is clear in the figures that our LSC algorithm is by far the better than the NJW algorithm. Our algorithm is very stable with the parameter k and always holds very good clustering results but the NJW algorithm is sensible to the parameter σ and it is very difficult to tune the parameter σ for NJW to achieve a good result. We can see clearly that LSC can correctly discover the underlying cluster structure.

3.2 Results on the Corel Data Set

We use the COREL image data set to evaluate our algorithm. In the experiments, we combine 64 dimensional color histogram and 64 dimensional Color Texture Moment (CTM [11]) to represent the images. More information about the data

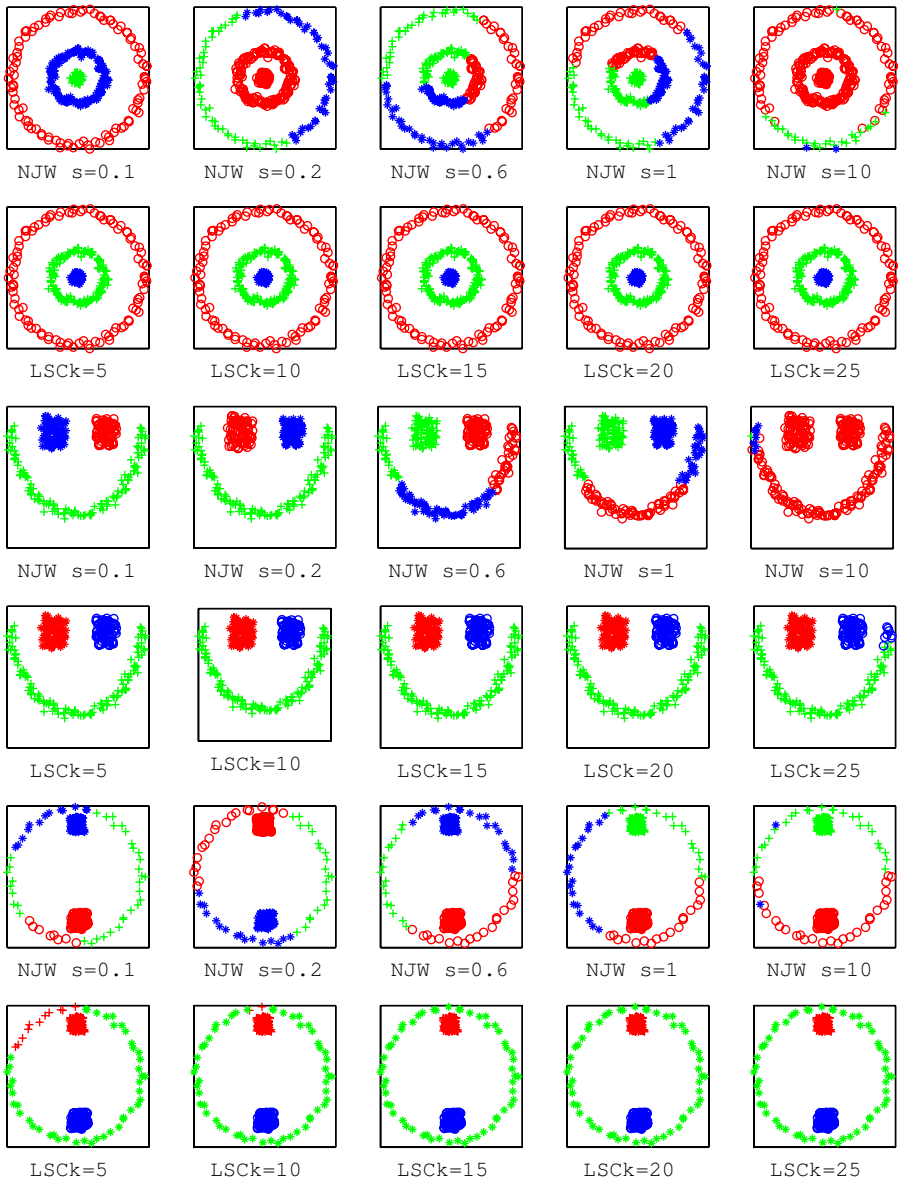
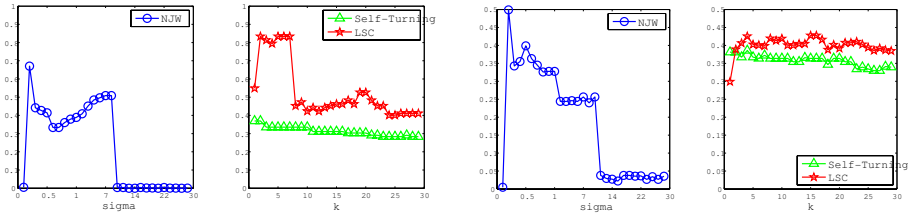
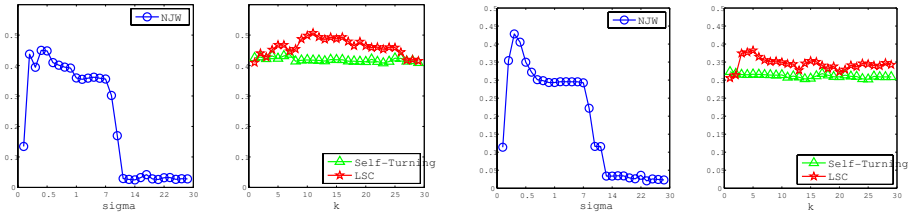


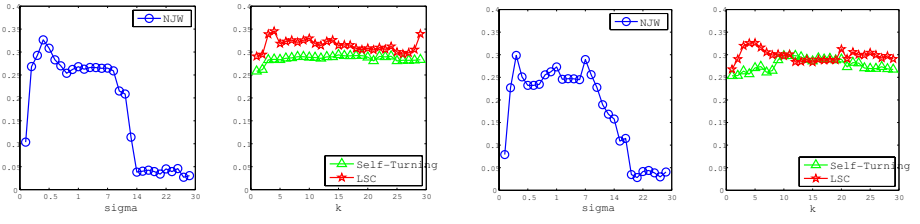
Fig. 1. Results on the toy data sets. The odd rows are results from the NJW algorithm and the even rows are results from our Lsc algorithm. For the NJW algorithm, we increase the parameter σ(s) from 0.1 to 10 respectively and for Lsc algorithm, we increase the parameter k from 5 to 25 respectively. From which we can see that the NJW algorithm us very sensible with the σ. The results also indicate that our algorithm is very stable with the parameter k and holds better accuracy.



(a) 2 clusters on COREL (b) 3 clusters on COREL



(c) 4 clusters on COREL (d) 5 clusters on COREL



(e) 6 clusters on COREL (f) 10 clusters on COREL

Fig. 2. The clustering results on the COREL data set

set can be find in [11]. From which 2 to 10 classes among 79 categories of color images were selected, where each class consists 100 images. We use the 1-2 class, 1-3 class, 1-4 class, 1-5 class, 1-6 class and 1-10 class to construct six data sets with different cluster numbers (2, 3, 4, 5, 6, 10) respectively. The results have been summarized in Fig. 2. The left sub-figure reports the results by the NJW and the right sub-figure reports the results by the LSC and Self-Tuning.The vertical axis of both sub-figures are set to the same for comparison.

In the experiments, he horizontal axis for the LSC and Self-Tuning algorithm the horizontal axis represents the parameter k from 2 to 30 [1] and for the NJW algorithm it represents the different parameter σ from 0.1 to 20. The vertical axis in all figures represents the *NMI*. In the experiments, it is clear that our LSC algorithm is by far the better than other algorithms. Our algorithm is very stable with the parameter k and always holds very good clustering results, but the NJW algorithm is very sensible to the parameter σ. It is very difficult to

[1] Here the k are start from 2 since the parameter 1 is too small for a connected graph.

optimize a good σ for NJW. Although Self-Turning is stable with parameter k, but it is not as good as LSC in clustering accuracy.

4 Conclusions

In this paper, we propose a novel spectral clustering algorithm called: Locality Spectral Clustering (LSC). The LSC algorithm firstly learn a smooth manifold structure on the data manifold and then computes the eigenvectors on the manifold, then the LSC algorithm use the solved eigenvectors to help the k-means algorithm to do clustering. Experiments have been performed on toy data sets and real world data sets and have shown that our algorithm is much better than former methods in accuracy and parameter stability.

References

1. von Luxburg, U.: A Tutorial on Spectral Clustering. In: Proc. British Machine Vision Conference. Statistics and Computing, vol. 17(4) (2007)
2. Scott, G.L., Longuet-Higgins, H.C.: Feature grouping by relocalisation of eigenvectors of the proximity matrix. In: Proc. British Machine Vision Conference (1990)
3. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
4. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS 2004 (2004)
5. Kannan, R., Vempala, S., Vetta, V.: On Spectral Clustering-Good, Bad and Spectral. In: FOCS (2000)
6. Zelnik-Manor, L., Perona, P.: Self-Tuning Spectral Clustering. In: NIPS (2004)
7. Wang, F., Zhang, C.: Label Propagation through Linear Neighborhoods. In: ICML (2006)
8. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)
9. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schlkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing System, vol. 16 (2004)
10. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), C2323–C2326 (2000)
11. Yu, H.-J.Z.H., Li, M., Feng, J.: Color texture moments for content-based image retrieval. In: International Conference on Image Processing (2002)
12. Cai, D., He, X., Han, J.: Document clustering using locality preserving indexing. IEEE Transactions on Knowledge and Data Engineering (2005)
13. He, X., Niyogi, P.: Locality preserving projections. In: Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge (2003)
14. Belkin, M., Niyogi, P., Sindhwani, V.: On Mani- fold Regularization. In: Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (2005)
15. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing System (2004)

Mining Arbitrarily Large Datasets Using Heuristic k-Nearest Neighbour Search

Xing Wu, Geoffrey Holmes, and Bernhard Pfahringer

Department of Computer Science
University of Waikato
Hamilton, New Zealand
{xw113,geoff,bernhard}@cs.waikato.ac.nz

Abstract. Nearest Neighbour Search (NNS) is one of the top ten data mining algorithms. It is simple and effective but has a time complexity that is the product of the number of instances and the number of dimensions. When the number of dimensions is greater than two there are no known solutions that can guarantee a sublinear retrieval time. This paper describes and evaluates two ways to make NNS efficient for datasets that are arbitrarily large in the number of instances and dimensions. The methods are best described as heuristic as they are neither exact nor approximate. Both stem from recent developments in the field of data stream classification. The first uses Hoeffding Trees, an extension of decision trees to streams and the second is a direct stream extension of NNS. The methods are evaluated in terms of their accuracy and the time taken to find the neighbours. Results show that the methods are competitive with NNS in terms of accuracy but significantly faster.

1 Introduction

The k-nearest neighbour search (kNN) algorithm is a very simple and effective algorithm. Unlike other classification algorithms it performs two roles simultaneously. First, given a test example it returns a classification or class probability distribution. Second, it returns the k-nearest neighbours to that example from its training set. Due to its widespread use in a number of fields, it has been ranked among the top ten most important algorithms in data mining [4].

The simplest way to find k-nearest neighbours is to memorize the entire training dataset and then given a test instance, linearly scan through the training dataset, using a distance function such as Euclidean distance to find a group of k instances in the training dataset which have the shortest distances to the test instance. The prediction for the test instance's class is simply the majority class of this group of training instances. The total time taken for classification and retrieval using this simple linear model of NNS is proportional to the product of the number of instances in both the training and testing sets with the number of dimensions (d), which causes the classification to become very slow, especially for large datasets with high dimensionality.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 355–361, 2008.

Appropriate data structures can be used to improve the performance of the basic algorithm, for example, KD-Trees, Metric-Trees, and Cover-Trees. Their performance for NNS has recently been compared empiricially by Kibriya and Frank [2]. They found that all these methods suffer from the the curse-of-dimensionality for $d > 16$, where their performance becomes worse than the brute force method.

Linear NNS is often referred to as an exact method. A variation of NNS permits the discovery of ϵ-approximate NN allowing an extra tolerance in the matches found. In this paper we adopt yet another variation which could be termed *heuristic*, in that we return neither exact nor ϵ-approximate nearest neighbours. Our neighbours perform at least as well as their exact counterparts in terms of accuracy but may be further away, on average, from the test instances.

This paper presents two heuristic methods to scale up NNS for independent and identically distributed datasets (iid) of large size and large dimensionality. Both stem from recent developments in the field of data stream classification. The first uses Hoeffding Trees, an extension of decision trees to streams and the second is a direct stream extension of NNS. The next section describes these two methods in detail. Section 3 presents an empirical evaluation against linear NNS. Section 4 concludes.

2 Speeding Up Nearest Neighbour Search

The demands of data stream processing have led to the development of a series of algorithms by Domingos and Hutlen based around the use of *Hoeffding bounds* [1]. The Hoeffding bound states that with probability $1 - \delta$, the true mean of a random variable of range R will not differ from the estimated mean after n independent observations by more than:

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}} \tag{1}$$

When building a classification data structure for a stream the bound can be used to make major decisions because the bound guarantees that the decision made at some time during data processing will not be significantly different from the one that would be made at the end, as long as the data is iid.

2.1 Using a Hoeffding Tree for Nearest Neighbour Search (HT-kNN)

As with other decision trees, *Hoeffding Trees* (HT) split on the attribute which has the highest information gain. The only difference is that *Hoeffding trees* use the *Hoeffding Bound* to decide when to split. The range R for the information gain is the base 2 logarithm of the number of possible class labels, and $1 - \delta$ is the user-defined confidence associated with the bound. With R and δ fixed, the only factor for the *Hoeffding bound* (ϵ) is the number of observations (n). Periodically, the information gain for each candidate's attributes is computed. If the difference in information gain for the top two attributes is more than ϵ,

then the tree is split on the attribute with the highest information gain [3]. This process continues until all the data has been observed. It is possible to then retrain the model by passing the training data through the tree again, several times in fact. This can lead to further structure being added, and consequently a more accurate tree may be formed.

A *Hoeffding tree* was built using the training data (data can be passed through the classifier one or multiple times by a user-defined number of passes, in this paper we only report results for a single pass). When training is finished, all the training instance indices were recorded as belonging to a particular leaf of the tree. Then for each test instance, we find the leaf it belongs to and we use linear NNS to find the k-nearest neighbours from the training data (we tried k equal to 20, 50, and 100) at each leaf. Thus, HT is used as a heuristic index structure to the training data in the same way that KD-Trees are used for exact NNS.

The *Hoeffding Tree* is being used as a preliminary lightweight data structure for kNN, reducing both the storage and time complexity of the operation. The efficacy of this method, rests on the ability of HT to divide similar instances into the leaves of the tree.

2.2 Using the Hoeffding Bound for Nearest Neighbour Search (HB-kNN)

Having established that the HT is a good structure for kNN, it seems natural to directly apply the bound to kNN using the methodology advocated in [1]. It is not at all obvious how to do this for a nearest neighbour algorithm because, as previously stated, it performs two roles simultaneously. The algorithm outlined below focusses on the predictive performance of the classifier not on the *goodness* of the nearest neighbours.

The algorithm for computing kNN using a Hoeffding bound is shown in Table 1. The basic idea is to incrementally compute the average class distribution (ACD) and use this as the basis for stopping the search. This occurs when the difference in probability for the most likely class and the next most likely class of the ACD is bigger than the Hoeffding bound. The intuition is that the difference in these probabilities will not change further irrespective of the amount of data we continue to process. While the ACD does not have a direct influence on the nearest neighbours it enables a representative set of training instances to be considered for inclusion in the nearest neighbour set. If the class distribution of the nearest neighbours is used to guide the search then it might exit immediately with the first k training instances, as these could easily contain a majority class by chance.

As with the Hoeffding tree, we do not need to check the stopping criterion after each training instance, rather it can be checked periodically using the user supplied grace period Δn. Finally, note that we use the probabilities of the classes rather than their counts. This is because we can then set the range in the HB calculation to one and this simplifies the computation for ϵ. Note that the method returns the k nearest neighbours from which the test instance's class value can be estimated by majority vote.

Table 1. The Hoeffding bound k Nearest Neighbour search algorithm

Given: Tr An iid training set.
Te A test instance
k Number of nearest neighbours
Δn Grace period
ACD Average class distribution

Procedure *HBkNNSearch(Tr, Te, k)*
Read first k instances from *Tr*.
Store k instances as the k nearest neighbours (kNN)
Compute class distribution ACD
n = k
For the remainder of instances X_i in *Tr*
n++.
If $X_i \in k$NN for *Te*
Update (kNN)
Update (ACD)
Every Δn
Find p_1,p_2: prob of the most likely and next most likely classes of ACD
If $(p_1 - p_2) > \epsilon$ then Break.
Return kNN

Finally, there is a problem with computing the confidence value δ. If we directly use the formula for ϵ, we cannot handle $1/\ \delta$ after the value for δ is less than 10^{-320}. If δ is made equal to 10^{-x} then it can be shown that the term under the square root in Equation 1 is proportional to x, and therefore δ can be made arbitrarily small by increasing the value of x.

3 Experimental Evaluation

We compared linear kNN, Hoeffding Trees with our implemented HT-kNN, and HB-kNN on large synthetic multi-dimensional datasets and real-world datasets from the UCI repository. All the experiments were carried out on machines with the same settings and memory limit. We used 20 nearest neighbours for all the experiments (experiments for 50 and 100 were conducted but are not reported, we comment on these later), and a default Grace period $\Delta n = 200$.

3.1 Datasets for Experiments

The datasets we used are listed in Table 2 below. Each of the datasets has one million instances. We randomized the datasets and split each of them into training and testing datasets. Note that it is not feasible to perform cross-validation evaluation with the linear kNN, hence the use of train-test splits. The split ratios of the datasets vary because they were calculated so that the running time of linear kNN for 20 nearest neighbours was about 10 hours (the actual times can be seen in Table 3, most are close to 600 minutes).

Table 2. The datasets used for the experiments

Name	Generator	Attributes	Training/test examples
RTS	Simple random tree	10 nominal,10 numeric	909090/ 90910
RTC	Simple random tree	50 nominal, 50 numeric	978260/ 21740
RRBFS	Random radial basis function (RBF)	Refers to a simple random RBF dataset 100 centers and 10 attributes	800000/ 200000
RRBFC	Random radial basis function	Refers to a complex random RBF dataset 1000 centers and 50 attributes	923076/ 76924
WAVE21	Waveform	21 numeric attributes, all include noise	941176/ 58824
WAVE40	Waveform	As above with an additional 19 irrelevant attributes	977777/ 22223

3.2 Comparison of kNN with HT, and HT-kNN

Table 3 below shows the accuracy and time usage for kNN, HT, and HT-kNN when k is set to 20, and $\delta = 10^{-7}$. The HT classifier does not return k nearest neighbours, it is used here simply to demonstrate that it can achieve a fast (the average speed was more than 90 times faster) and accurate (the average accuracies are similar, 83.2% for linear kNN and 82.9% for HT) division of data. The algorithm HT-kNN is a good compromise in that it is approximately 33 times faster than linear kNN on average and more accurate (83.2% for linear kNN against 87.6% for HT-kNN). These results are somewhat biased by the first two datasets where HT solutions should perform well. These datasets were generated by a Random Tree Generator, which should favour the decision tree learner. We can see that except for the first two datasets the nearest neighbour search always has a higher accuracy than the HT classifier. However, applying nearest neighbour to the leaves narrows the gap. In fact, the gap can be closed entirely if cross-validation is used to discover the best number of nearest neighbours to use at each leaf (rather than a fixed number). This is shown in the final column, overall using cross-validation increases accuracy (to 88.3% on average) but takes away much of the time advantages afforded by the tree (HT-cross validated is only three times quicker than linear kNN). Of course, if we had 100 times more test instances the speedup could still be a lot more than threefold, this factor is only true for the specific train/test setup we use here. Although results are not presented, the pattern is similar for k equal to 50 and 100.

3.3 Comparison of kNN with HB-kNN

HB-kNN can always attain the same accuracy as kNN as long as the δ value is small enough. As the threshold is adjusted (i.e. the value of x is increased to

Table 3. Classification rate and time cost for kNN and HT with kNN

Dataset	20 NN Accuracy / Time	HT Accuracy / Time	HT- 20NN	HT-cross validation
RTS	81.91 575m54.67s	97.64 7m39.96s	97.99 23m47.73s	98.00 152m14.95s
RTC	53.55 596m6.10s	69.22 3m28.24s	69.05 6m1.53s	69.88 121m30.52s
RRBFS	94.35 593m49.82s	85.88 15m27.38s	92.23 20m43.85s	92.63 109m21.21s
RRBFC	99.99 632m16.72s	86.88 5m57.32s	97.99 20m28.95s	99.50 169m13.00s
WAVE21	85.35 602m50.74s	78.94 6m45.10s	84.53 24m22.07s	85.06 266m56.04s
WAVE40	84.58 565m17.83s	78.90 3m59.54s	83.60 15m10.17s	84.40 446m35.53s

Table 4. Classification rate and time for kNN and HB-kNN

Dataset	Accuracy	δ value	Time for kNN	Time for HB-kNN
RTS	81.91	10^{-5000}	575m54.66s	110m6.91s
RTC	53.55	10^{-100}	596m6.10s	11m44.03s
RRBFS	94.35	10^{-25000}	593m49.82s	94m12.32s
RRBFC	99.99	10^{-15000}	632m16.72s	91m54.96s
WAVE21	85.35	10^{-100}	602m50.74s	3m59.40s
WAVE40	84.58	10^{-200}	565m17.83s	3m22.43s

make δ smaller), the accuracies get closer and closer to the performance of linear kNN. Of course, this means we need to go through more and more instances in the training set, and this means more time to classify the data.

Table 4 shows the baseline accuracy for linear kNN, the δ value for HB-kNN, and time cost for both kNN and HB-kNN. The time cost for HB-kNN to achieve the same accuracy has a range from 3 to 110 minutes. This cost is clearly problem dependent. For example, RRBFS and RRBFC generate a fixed number of random centroids with random positions, standard deviations, class labels and weights assuming a Gaussian distribution. When a new example is generated, it randomly selects a centroid, taking weights into account, and the label for this example will be the same as the centroid. This creates a normally distributed hypersphere surrounding each central point with varying densities. Therefore, when we use HB-kNN, we read in 200 (the grace period) iid samples each time from the training set and find the 20 nearest neighbours. When the first 200 training instances come in, they could belong to any hypersphere. So when the classification is done it could easily misclassify the test instance. When the next 200 instances come in, it increases the chance of the test instance being classified

correctly, because the training instances which belong to the same hypersphere will have shorter distances, so they will be added to the nearest neighbours to replace the ones which come from another hypersphere. As we know that the training dataset is iid, there is likely to be a roughly equal probability of reading training instances from each hypersphere. Thus, we need a very small δ value, i.e. to see a very large amount of training instances to achieve the baseline accuracy.

4 Conclusions

This paper introduced HT-kNN, HT-cross validation, and HB-kNN methods to allow kNN to be applied to datasets with high dimensions. Of these methods, HT-kNN is the fastest, on average it is 90 times quicker than linear kNN. Using cross validation to select the number of nearest neighbours to use at each leaf improves accuracy slightly but is much slower. The accuracy of these methods depends heavily on the structure of the dataset. Not surprisingly, HT-kNN and HT-cross validation have better accuracies on datasets that favour decision trees, for other datasets accuracies can be equivalent but only if cross validation is used.

When the δ value is small enough, the accuracy of HB-kNN can always match the linear kNN, and the matching δ depends on the structure of the dataset. As the δ value decreases (the confidence increases) the average neighbourhood distance for HB-kNN decreases, at the same time as the similarity increases.

References

1. Domingos, P., Hulten, G.: Mining complex models from arbitrarily large databases in constant time. In: international conference on Knowledge discovery and data mining, pp. 525–531 (2002)
2. Kibriya, A.M., Frank, E.: An Empirical Comparison of Exact Nearest Neighbour Algorithms. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 140–151. Springer, Heidelberg (2007)
3. Kirkby, R.: Improving Hoeffding Trees, PhD thesis, University of Waikato (2007)
4. Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowledge and Information Systems 14, 1–37 (2007)

Cross-Domain Knowledge Transfer Using Semi-supervised Classification

Yi Zhen[1] and Chunping Li[2]

[1] Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, China
yzhen@cse.ust.hk
[2] School of Software, Tsinghua University
Beijing 100084, China
cli@tsinghua.edu.cn

Abstract. Traditional text classification algorithms are based on a basic assumption: the training and test data should hold the same distribution. However, this identical distribution assumption is always violated in real applications. Due to the distribution of test data from target domain and the distribution of training data from auxiliary domain are different, we call this classification problem cross-domain classification. Although most of the training data are drawn from auxiliary domain, we still can obtain a few training data drawn from target domain. To solve the cross-domain classification problem in this situation, we propose a two-stage algorithm which is based on semi-supervised classification. We firstly utilizes labeled data in target domain to filter the support vectors of the auxiliary domain, then uses filtered data and labeled data from target domain to construct a classifier for the target domain. The experimental evaluation on real-world text classification problems demonstrates encouraging results and validates our approach.

1 Introduction

One common problem in the information era is the vast amount of unorganized data. Almost everyday, people deal with data classification, especially document classification, for example, classifying emails, reports, etc. Document classification has been a hot research area for a long time and still have many interesting topics.

Traditional document classification methods are based on the *identical assumption*: the training and test sets are drawn from the same distribution. However, this assumption is often violated in real applications, that is, the two sets are drawn from related but different distributions. Let's take anti-spam classifier for example: given sufficient emails labeled spam or non-spam as training set, we can train an accurate classifier on the training set. At first, the classifier performs quite well, but the performance may degrade after several months. The reason why the classifier cannot classify new emails as accurate as before is that the

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 362–371, 2008.

new emails are intrinsically obey another distribution which is quite different from the training set. Though the training set is less useful, it is not wise to discard all the old labeled data. If we label a few data in target domain and use them to select the useful data in the old training set, and then use both of the data sets as training set, we may get a better classifier than before.

In this paper, we propose a novel support vector based semi-supervised algorithm to solve this problem. First, we get the support vectors of the training set drawn from the auxiliary domain. Then, we do semi-supervised classification on a few labeled data in target domain and the support vectors obtained in the first step. After getting the predicted labels of the support vectors, we pick the support vectors which are correctly predicted as high-confident data points of the training set. At last, we use both the few labeled data from target domain and the high-confident data points from auxiliary domain to train a classifier for the unlabeled data in the target domain.

The remainder of the paper is organized as follows. We firstly review the previous works related to cross-domain text classification in Section 2. Section 3 presents our proposed two-stage algorithm, which is based on filtering support vectors. The experimental results are shown in Section 4. In Section 5, we conclude the paper and give out some future works.

2 Related Work

2.1 Transfer Learning

Transfer learning is what happens when we use knowledge learned from one domain to get knowledge from another domain. It happens in many aspects of our life even since we were infants. For example, told what is the difference between an apple and a basketball, a little child can easily tell an pear from an football. Obviously, knowledge can be transferred between related domains although there are something different. However, the mechanism behind the transfer process is still unknown.

Recently, transfer learning has been recognized as an important topic in research areas such as machine learning and data mining. Transfer learning was originated in multi-task learning whose objective is to discover common knowledge in multiple tasks. Some early works can be found in [1][2][3][4][5]. Ben-David[6] and Ando[7] have theoretically justified transfer learning in multi-task learning respectively.

So far, there has been no clear definition of transfer learning. However, the learning process does exist. Many people published papers in this domain but with different keywords or task names, such as sample selection bias[8], class imbalance problem[9], concept or target drift[10].

2.2 Semi-supervised Learning

Traditional supervised machine learning methods have a basic assumption that the labeled data are sufficient. However, in real applications, it is often difficult

to get labeled data, while unlabeled data may be much easier to obtain. Semi-supervised learning addresses this problem by using both labeled and unlabeled data to train a classifier and predict the labels of the unlabeled data. There are two main assumptions in semi-supervised learning: smoothness assumption and cluster assumption. Smoothness assumption means if two points in a high-density region are close, they should belong to the same class. Cluster assumption means if points are in the same cluster, they are likely to belong to the same class. See the latest survey [11] for more details.

Here we focus on semi-supervised classification. Many semi-supervised classification algorithms have been applied on text collections, including transductive support vector machines[12], graph-based algorithms[13][14], expectation maximization[15] and co-training[16], etc.

3 Two-Stage Algorithm for Cross-Domain Text Classification

Cross-domain text classification is an interesting and open problem. We first give out the formal definition of cross-domain classification.

In traditional text classification problem, we let X be the instance space, which means the set of all documents and $Y = \{+1, -1\}$ be the set of labels. A classifier is a map function from X to Y. Generally, we train a classifier on the training set, which is represented by $T \subseteq \{X \times Y\}$ and then apply it on the test set $S = \{x_i\}$, where $x_i \in X$, to get the labels of S.

However, in the problem of cross-domain text classification, there are two types of instance spaces, X_t and X_a. X_t is the target instance space, in which the documents obey the distribution of target domain; X_a is the auxiliary instance space, in which the document obeys the distribution of auxiliary domain. The training data is as well partitioned into two parts, which are *target training data* $T_t \subset X_t$ and *auxiliary training data* $T_a \subset X_a$. The test data is denoted by $S = \{x_i\}$, where $x_i \in X_t$. Usually, the size of T_t is much smaller than that of T_a, which means $|T_t| << |T_a|$.

Dai et al. proposed co-clustering based algorithm[17], which build word clusters to bridge the gap between the two domains to handle this problem. They also transfer Naive Bayes Classifier[18] for this purpose. However they assume the labeled data in test domain are sufficient in the above two methods. They proposed a Tradaboost algorithm[19], which extended the Adaboost algorithm, to get an accurate classifier by iteratively updating the weights of data points when training a SVM classifier.

To transfer knowledge from auxiliary domain to target domain, we need to clarify which parts of knowledge learned from auxiliary domain are suitable for target domain. For this purpose, we make use of a few labeled data drawn from target domain. More specifically, we call these training data *target training data*. However, the quantity of target training data is not sufficient to train a good classifier. The other set of training data ,whose distribution is different from target domain, is called *auxiliary training data*. The quantity of auxiliary data

is adequate, but classifiers trained from these data cannot classify test data well due to difference between the two distributions.

Actually, our goal is to find the useful data points in T_a, which can help to classify data in target domain. As the label sets in both domain are the same, it is natural to select the data points near the decision surface of T_a as the most important data points. Support vector machine is justified to select data points which can maximize the margin between two classes. If all the support vectors are found, other data points are redundant. However, the decision surface of target domain and auxiliary domain are different, so we should find a method to select the effective support vectors in both domains. To solve this problem, we use T_t to select the effective support vectors of T_a, which are more likely to be support vectors of target domain documents than others. After the selection, we can construct a new training set with higher quality.

The details of our algorithm is shown in Algorithm 1:

Algorithm 1. Cross-Domain Classification Algorithm(CDC)

Input: the two labeled data set T_t and T_a, the unlabeled data set S.
output: a classifier
Procedure:
step 1: train a standard SVM classifier on T_a and get set of support vectors SV in T_a.
step 2: take SV as unlabeled data and use T_t to do semi-supervised classification and predict the labels of SV.
step 3: select the support vectors in SV which are correctly predicted in *step 2* as SV^*.
step 4: train a SVM classifier on T_t and SV^* to get a classifier.

4 Experiments

In this section, the performance of our algorithm on text classification is shown. Here we focus on the binary classification, but it is easy to extend the binary classifier to multi-class classifier.

In the experiments, we use *rainbow*[20] to preprocess the documents in 20 Newsgroups. After stemming and removing stop words, we eliminate the words appeared in less than 6 documents. We also cut off the headers of documents because the categories are presented in the headers.

To implement our algorithm, we use the toolkit of SVM^{light}[21] to get the support vectors in step 1 and the final classifier in step 4. We also use TSVM[12] as semi-supervised classifier for step 2.

4.1 Data Preparation

We conduct the experiments on 20 Newsgroups[1] data set, which is widely used for data mining and machine learning research. It contains nearly 20,000 newsgroup documents which are grouped into 20 low-level groups and 6 high-level

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 1. Construction of Data Sets for Cross-Domain Text classification

Data Set	D_t	D_a
comp vs sci	comp.sys.ibm.pc.hardware	comp.graphics
	comp.sys.mac.hardware	comp.os.ms-windows.misc
	comp.windows.x	sci.crypt
	sci.med	sci.electronics
	sci.space	
comp vs rec	comp.os.ms-windows.misc	comp.graphics
	comp.windows.x	comp.sys.ibm.pc.hardware
	rec.autos	comp.sys.mac.hardware
	rec.sport.baseball	rec.motorcycles
		rec.sport.hockey
comp vs talk	comp.os.ms-windows.misc	comp.graphics
	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware
	talk.politics.guns	comp.windows.x
	talk.politics.misc	talk.politics.mideast
		talk.religion.misc
rec vs sci	rec.motorcycles	rec.autos
	rec.sport.hockey	rec.sport.baseball
	sci.crypt	sci.med
	sci.electronics	sci.space
rec vs talk	rec.sport.baseball	rec.motorcycles
	rec.sport.hockey	rec.autos
	talk.politics.mideast	talk.politics.guns
	talk.religion.misc	talk.politics.misc
sci vs talk	sci.crypt	sci.electronics
	sci.space	sci.med
	talk.politics.guns	talk.politics.misc
	talk.politics.mideast	talk.religion.misc

groups. Here we only use 4 high-level groups, including *sci*, *talk*, *computer* and *rec*, to construct cross-domain data sets. Details are provided in the following subsection.

Cross-domain Data Sets. We construct six different data sets for cross-domain classification. As we can see from Table 1, the class label of each document is determined by the high-level category. So the class labels in both domains are the same, though the low-level categories are different in different domains. There are two domains, one is *target domain* which is represented by D_t, and the other is *auxiliary domain* which is represented by D_a. Our goal is to classify the documents in D_t with the knowledge learned from labeled data in the D_a.

We get test set S from the target domain and two training sets T_t and T_a from D_t and D_a respectively.

Difference between Domains. We use *Kullback-Leibler (K-L) divergence*[22] to measure the difference between distributions of *target domain* and *auxiliary domain*.

Table 2. Description of Cross-Domain Data

Data Set	$\|S\|$	$\|T_t\|$	$\|T_a\|$	K-$L(\ T_t)$	K-$L(\ T_a)$
rec vs talk	1406	2120	2183	0.425	1.386
comp vs sci	1936	2923	2343	0.424	1.370
rec vs sci	1576	2366	2360	0.474	1.185
sci vs talk	1515	2293	2010	0.431	1.015
comp vs rec	1559	2347	2914	0.934	0.980
comp vs talk	1450	2172	2669	0.884	0.967

The *K-L divergence* between two probability mass functions $p(x)$ and $q(x)$ is defined by Equation 1.

$$K\text{-}L(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{1}$$

So the *K-L divergence* between the two text domains (*target domain* D_t and *auxiliary domain* D_a)is defined by Equation 2.

$$K\text{-}L(D_t||D_a) = \sum_w \Pr(w|D_t) \log \frac{\Pr(w|D_t)}{\Pr(w|D_a)} \tag{2}$$

In Equation 2, $\Pr(w|D_t)$ and $\Pr(w|D_a)$ represents the probability of occurrence of word w in D_t and D_a respectively. Let $occ(w, D_i)(i = \{t, a\})$ be the occurrence count of word w in domain D_i, $\Pr(w|D_t)$ and $\Pr(w|D_a)$ can be computed by Equation 3 and Equation 4. We use laplace smoothing technique in Equation 3 and Equation 4, in which $|W_i|(i = \{t, a\})$ stands for the vocabulary size of $D_i(i = \{t, a\})$.

$$\Pr(w|D_t) = \frac{1 + occ(w, D_t)}{|W_p| + \sum_w occ(w, D_t)} \tag{3}$$

$$\Pr(w|D_a) = \frac{1 + occ(w, D_a)}{|W_a| + \sum_w occ(w, D_a)} \tag{4}$$

Table 2 gives the description of the data sets we used. The columns 2-4 are the documents count of test set S, training set from target domain T_t and training set from auxiliary domain T_a. the last two columns are the *K-L divergence* between S and T_t, and *K-L divergence* between S and T_a.

Table 3 shows the comparison of error rate of normal SVM classifiers trained on target domain and auxiliary domain and tested on target domain. Obviously, the classifier trained on T_a has much poorer accuracy than the classifier trained on T_t.

4.2 Experimental Results

To validate the effectiveness of our algorithm, we conducted three groups of experiments which are presented in details in the following subsections.

Table 3. Error Rate of Classifier Trained on Target and Auxiliary Domain

	T_t	T_a
rec vs talk	0.013	0.336
comp vs sci	0.029	0.267
rec vs sci	0.025	0.176
sci vs talk	0.029	0.211
comp vs rec	0.023	0.108
comp vs talk	0.018	0.037

Table 4. Performance of different training sets, $|T_t| = 20$ $|T_a| = 2000$

Data Set	T_t	T_a	$T_t + T_a$	$T_t + SV$	$T_t + SV^*$	Error Rate Reduction
rec vs talk	0.2910	0.3830	0.3178	0.3171	0.2774	0.1271
comp vs sci	0.4239	0.2657	0.2534	0.2530	0.1933	0.2372
rec vs sci	0.3349	0.1811	0.1634	0.1638	0.1518	0.0710
sci vs talk	0.4401	0.2116	0.1917	0.1905	0.1688	0.1195
comp vs rec	0.2992	0.0769	0.0716	0.0718	0.0663	0.0740
comp vs talk	0.3938	0.0305	0.0296	0.0296	0.0321	-0.0845

Performance on Training Sets. First of all, we compare the error rates of classifiers trained from different training sets, including training set from target domain(T_t), training set from auxiliary domain(T_a), training set from both domains($T_t + T_a$), training set from target domain and support vectors of T_a ($T_t + SV$), training set from target domain and filtered support vectors of T_a ($T_t + SV^*$).

In the experiments, we randomly select 20 documents from target domain as T_t and 2000 documents from auxiliary domain as T_a. We conduct each experiments 10 times and present average values.

As we can see from Table 4, the error rate of SVM classifier trained on $T_t + SV^*$ is much lower than that of other training sets. The last column shows the error rate reduction of $T_t + SV^*$ to $T_t + T_a$. According to the decrease of the value of *K-L divergence*, which means the target domain and auxiliary domain are more and more similar, the reduction is decreasing. Note that the error rate of $T_t + T_a$ and $T_t + SV$ are almost the same in every data set, which means that support vectors are really the most important data points to the classification task.

Size Reduction. In this subsection, we compare the size of T_a, SV and SV^*. All of the numbers are the averages of 10 experiments. As we can see from Table 5 about 30% support vectors in SV and 70% data points in T_a are discarded by our algorithm. This result proved that many data points in T_a are useless for the classification of documents in target domain. We should note that in data sets "comp vs rec" and "comp vs talk", only 15% support vectors are filtered out. That is because in these two data sets, the target domain and auxiliary domain are much similar than other cases and semi-supervised classification would do less help.

Table 5. Size Reduction of Training Sets

Data Set	$\|T_a\|$	$\|SV\|$	$\|SV^*\|$	1-$\|SV^*\|/\|SV\|$	1-$\|SV^*\|/\|T_a\|$
rec vs talk	2000	1017.3	747.6	0.2651	0.6262
comp vs sci	2000	1096.6	728.1	0.3360	0.6360
rec vs sci	2000	1104.9	714	0.3538	0.6430
sci vs talk	2000	1101.9	713.4	0.3526	0.6433
comp vs rec	2000	961.8	804.1	0.1640	0.5980
comp vs talk	2000	959.2	839	0.1253	0.5805

Table 6. Performance of Classifiers Trained on different size of T_t

$\|T_t\|/\|T_a\|$	T_t	$T_t + T_a$	$T_t + SV$	$T_t + SV^*$	Error Rate Reduction
1%	0.2910	0.3178	0.3171	0.2774	0.1271
2%	0.2526	0.2201	0.2181	0.1698	0.2285
3%	0.1293	0.1704	0.1676	0.1189	0.3022
4%	0.0682	0.1326	0.1294	0.0912	0.3122
5%	0.0816	0.1042	0.1021	0.0781	0.2505
6%	0.0598	0.0890	0.0872	0.0690	0.2247
7%	0.0423	0.0743	0.0735	0.0580	0.2194
8%	0.0418	0.0693	0.0679	0.0544	0.2150
9%	0.0332	0.0622	0.0600	0.0513	0.1752
10%	0.0339	0.0546	0.0541	0.0469	0.1410

Target Training Sets with Different Size. In the last experiment, we compare the error rates of classifiers when the size of T_t varies. We use "rec vs talk " for comparison, because this data set has the largest value of *K-L divergence*. There are 2000 documents in T_a, and the size of T_t varies from 20 to 200. The average values of 10 experiments are shown in Table 6.

The last column shows the reduction of error rates between $T_t + SV^*$ and $T_t + T_a$. We can see that the error rate reduction first increases and then decreases. Because when the size of T_t is too small to train a classifier, it is not suitable to filter SV, get SV^* from auxiliary domain and then get enough data to train a classifier. However, when the size of T_t is so large that it is enough to train a classifier, SV^* from auxiliary domain may helps little. So our algorithm can be applied in situations with rather few labeled data in target domain.

5 Conclusions

In this paper, we proposed a two-stage cross-domain classification algorithm which is based on filtering support vectors through semi-supervised classification. The basic idea of our algorithm is to select useful data points in auxiliary domain to construct an additional training set. As we can see from the experimental results, our algorithm showed much lower error rate than directly using

training data in the auxiliary domain. So in this way, knowledge learned from the auxiliary domain can be transferred to the target domain. Due to its simplicity and effectiveness, our algorithm can be applied easily and handle large problems.

Note that in Step 2 of our algorithm, we use Transductive Support Vector Machines (TSVM) to do semi-supervised classification. TSVM is only one kind of the popular semi-supervised learning methods and based on low density separation. In future, we will try other methods such as graph-based algorithms for semi-supervised classification to see if they can get better performance.

Acknowledgement

This work was supported by National Nature Science Funding of China under Grant No. 90718022.

References

1. Caruana, R.: Multitask learning. Machine Learning 28(1), 41–75 (1997)
2. Thrun, S.: Is Learning the n-th thing any easier than learning the first? In: Advances in Neural Information Processing System (NIPS), pp. 640–646 (1996)
3. Baxter, J.: A Baysian/Information theoretic model of learning to learn via multiple task sampling. Machine Learning 28(1), 7–39 (1997)
4. Thrun, S., Mitchell, T.: Learning one more thing. In: Proceedings of 14th International Joint Conference on Artificial Intelligence (IJCAI 1995), pp. 1217–1223 (1995)
5. Schmidhuber, J.: On Learning How to Learn Learning Strategies. Technical Report FKI-198-94, Fakultat fur Informatik (1994)
6. Ben-David, S., Schuller, R.: Exploiting task relatedness for multiple task learning. In: Proceedings of 16th Annual Conference on Learning Theory (COLT 2003), pp. 567–580 (2003)
7. Ando, R., Zhang, T.: A Algorithm for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. Journal of Machine Learning Research (JMLR) 6, 1817–1853 (2005)
8. Zadrozy, B.: Learning and Evaluating Classifiers under Sample Selection Bias. In: Proceedings of 21th International Conference on Machine Learning (ICML 2004), pp. 114–121 (2004)
9. Elkan, C.: The Foundations of Cost-sensitive Learning. In: Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), pp. 239–246 (2001)
10. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23(1), 69–101 (1996)
11. Zhu, X.: Semi-Supervised Learning Literature Survey (2007), `http://pages.cs.wisc.edu/j~erryzhu/research/ssl/semireview.html`
12. Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines. In: Proceedings of the 16th International Conference on Machine Learning (ICML 1999), pp. 200–209 (1999)

13. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In: Proceedings of the 20th International Conference on Machine Learning (ICML 2003), pp. 912–919 (2003)
14. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing System (NIPS), vol. 16, pp. 321–328 (2004)
15. Nigam, K., Mccallum, A., Thrun, S., Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning 39(2-3), 103–134 (2000)
16. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT 1998), pp. 92–100 (1998)
17. Dai, W., Xue, G., Yang, Q., Yu, Y.: Co-clustering based Classification for Out-of-domain Documnets. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007), pp. 210–219 (2007)
18. Dai, W., Xue, G., Yang, Q., Yu, Y.: Transfering Naive Bayes Classifiers for Text CLassification. In: Proceedings of Association for the Advancement of Artificial Intelligence (AAAI 2007), pp. 540–545 (2007)
19. Dai, W., Yang, Q., Xue, G., Yu, Y.: Boosting for Transfer Learning. In: Proceedings of 24th International Conference on Machine Learning (ICML 2007), pp. 193–200 (2007)
20. McCallum, A.: Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering (1996), `http://www.cs.cmu.edu/~mccallum/bow`
21. Joachims, T.: Making large-Scale SVM Learning Practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge (1999)
22. Kullback, S., Leibler, R.: On Information and sufficiency. Annals of Mathematical Statistics 22(1), 79–86 (1951)

On the Limitations of Scalarisation for Multi-objective Reinforcement Learning of Pareto Fronts

Peter Vamplew, John Yearwood, Richard Dazeley, and Adam Berry

School of ITMS, University of Ballarat,
University Drive, Mt Helen, Ballarat, Victoria, Australia
{p.vamplew,j.yearwood,r.dazeley}@ballarat.edu

Abstract. Multiobjective reinforcement learning (MORL) extends RL to problems with multiple conflicting objectives. This paper argues for designing MORL systems to produce a set of solutions approximating the Pareto front, and shows that the common MORL technique of scalarisation has fundamental limitations when used to find Pareto-optimal policies. The work is supported by the presentation of three new MORL benchmarks with known Pareto fronts.

Keywords: multiobjective, reinforcement learning, scalarisation, Pareto fronts.

1 Introduction

Most reinforcement learning (RL) algorithms aim to maximise performance on a single objective. Many problems naturally fit this model, but there has been growing recognition in the optimisation community that many real-world problems exhibit multiple objectives [1], and that specialised multiobjective optimisation (MOO) techniques are required for these problems. Recently there has been interest in developing RL methods to handle multiobjective tasks. This paper argues that so far most multiobjective reinforcement learning (MORL) research has failed to capitalise on the knowledge in the MOO literature. Specifically this paper discusses the role of Pareto dominance within MORL, and examines the limitations of scalarised MORL.

2 Reviewing Existing Approaches to MORL

The easiest way to apply RL algorithms to multiobjective problems is to convert the problems themselves into single-objective tasks. In single-objective RL the reward is scalar, whereas in MORL it is a vector with an element for each objective. So a multiobjective task can be reduced to a single objective via scalarisation, which applies a function to the reward vector to produce a scalar reward. Commonly this is a linear weighted sum of the individual rewards [3, 4]. The weights allow the user some control over the nature of the solution, by placing greater or lesser emphasis on each objective. Less frequently a non-linear function tuned to the problem domain may be used [2]. In the simplest implementation, rewards are scalarised prior to reaching the agent, allowing the learning algorithm to remain unaltered. Alternatively the algorithm may be modified to learn the expected values for each objective, which may facilitate re-use of earlier learning when multiple policies are being found [3].

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 372–378, 2008.

A small number of alternatives to scalarisation have also been proposed. [5] assumes a known partial ordering of the objectives, and threshold values which must be achieved for objectives (e.g. a robot maintaining a non-zero energy level whilst accomplishing some task). This approach has been applied in the specific context of risk-sensitive learning [6, 7]. [8, 9] describe MORL algorithms to achieve long-term average rewards lying within an externally defined 'target' region in objective space. These produce non-stationary policies where actions are influenced by the current state and by the position of the current average reward relative to the target region.

These existing MORL systems find single solutions, whereas most MOO systems aim to produce a set of solutions which form a range of good compromises between the objectives. A 'good' compromise is often defined in terms of Pareto dominance. A solution dominates another if it is superior on at least one objective, and at least equal on all others. They are incomparable if each is superior on at least one objective (see Fig 1, which assumes the aim is to maximise each objective's value). A dominated solution is of little value, as the dominating solution is preferable. If all dominated solutions are eliminated from the set of all possible solutions, the resulting set is the globally optimal set of compromise solutions, known as the Pareto optimal front (see Fig 2). Of course finding the true front for any substantial problem is impractical, and so the goal is to produce a set of solutions which approximates the true front.

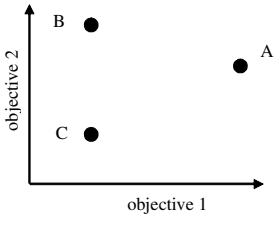


Fig. 1. Solutions A and B dominate C; solutions A and B are incomparable

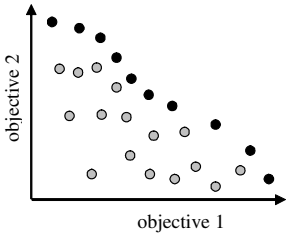


Fig. 2. Black points form the Pareto front; all grey points are dominated by at least one black point

There are several advantages to producing a set of solutions rather than a single solution. Methods producing a single solution require *a priori* input from the user about the desired nature of that solution (e.g. defining the partial-ordering and thresholds, or specifying objective weights). This requires domain knowledge on the part of the user, and minor variations may result in significant changes in the solution achieved (e.g. a slightly higher threshold for one objective may prevent discovery of a solution which provides a significant improvement on all other objectives). Systems which produce sets of solutions support *a posteriori* decisions about the accepted solution, which are better informed as they are based on knowledge of the trade-offs available as encapsulated by the front. Also the presentation of the front to the user may provide better insight into the relationships between the objectives. The primary disadvantage of generating multiple solutions is the increased computational cost and, for on-line learning, the increased time spent interacting with the environment.

As noted earlier, most existing MORL systems produce single solutions, but some authors have investigated generating multiple solutions. [10] describes a policy-gradient

MORL algorithm. A policy derived by applying RL independently to each objective, is improved using hill-climbing to follow gradients in the policy space which are non-negative with regards to all objectives. An approximate front is constructed by performing repeated searches with different weightings of the gradients. Whilst this approach is quite sophisticated, most other work on multiple-solution MORL has relied on the simpler combination of scalarisation and TD methods. [4] used a combination of scalarisation and Q-learning to approximate the Pareto front for a lake regulation system with twin objectives of ensuring water supply and providing flood protection. The front is found by repeated runs of the algorithm with varying objective weights. [3] applied scalarised RL to tasks where an external source of the objective weights was assumed, with periodic changes in these weights. The task of the system was to rapidly adapt to the novel weights. The key finding was that performance could be substantially improved by storing and re-using learning from earlier policies. Whilst this work did not directly consider approximating the Pareto front, it could be used for that purpose by replacing the external weight source with a loop which steps through the scalarisation weight space.

Unfortunately, whilst scalarised MORL is simple, it suffers from a fundamental flaw. Any system based on a linear combination of the objectives is incapable of finding solutions which lie in a concave region of the Pareto front [11]. No weights exist which allow a point in a concavity to produce a weighted sum higher than that achieved by the solutions at either edge of the concavity. As many multiobjective problems exhibit non-convex regions, this limitation has significantly reduced interest in scalarisation-based methods in the MOO literature [12]. The following sections of this paper will examine the significance of this limitation in the context of MORL.

3 MORL Benchmarks and Scalarisation Performance

Given the known limitations of scalarisation, why does scalarised MORL continue to be used? We argue this arises from the lack of any means for assessing the performance of a MORL system. MORL research is in its infancy, and no study has yet examined the performance of different algorithms – in fact no standard benchmarks have been established to act as a basis for such a study. In addition, the Pareto fronts were not known for the problems to which scalarised MORL has been applied. Known fronts can provide a measure of the absolute performance of an MORL system by comparing the approximate front produced against the true front. In the absence of such known fronts, it is difficult to judge the quality of the solutions produced by a learning system. Here we address these issues by presenting three benchmark tasks, along with their Pareto fronts. To our knowledge these are the first MORL tasks with known fronts. The first problem was designed specifically for this research. The second and third problems have been drawn from the single-objective RL literature and adapted to create multiobjective tasks. To facilitate future use, full details of the tasks and the data-points describing the fronts are available for download from http://uob-community.ballarat.edu.au/~pvamplew/MORL.html.

Deep Sea Treasure. This task consists of a grid of 10 rows and 11 columns (see Fig 3). The agent controls a submarine searching for treasure. There are multiple

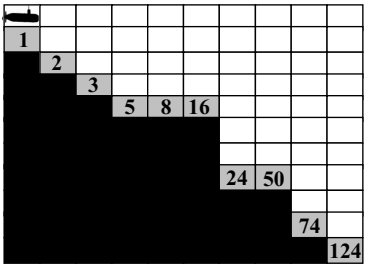


Fig. 3. Deep Sea Treasure: Black cells are the sea-floor; grey cells are treasure locations

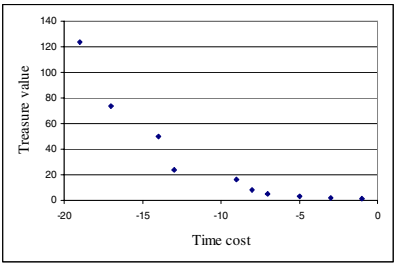


Fig. 4. The Pareto front for the Deep Sea Treasure problem

treasures with varying values. There are two objectives – to minimise the time taken to reach treasure, and to maximise the value of the treasure retrieved. This is an episodic task – each episode starts with the vessel in the top row of the first column, and ends when a treasure is reached or after 1000 actions. At each time-step, four actions are available to the agent – moving one square to the left, right, up or down. Any action which would result in the agent leaving the grid will leave its position unchanged. The reward received by the agent on each turn is a 2-element vector. The first element is a time penalty, which equals -1 on all turns. The second element is 0 on all turns except when the agent moves into a treasure location, when it is the value shown in Fig 3. There are ten non-dominated policies, each of which leads to one of the ten treasure locations. The Pareto front of these policies is shown in Fig 4. The front is globally concave, with local concavities at the second, fourth and sixth points from the left.

MO-Puddleworld. This is a 2-D environment. The agent starts each episode at a random state and must reach the goal in the top-right corner, whilst avoiding puddles. It receives its current coordinates as input, and at each step selects an action (left, right, up or down). The agent's position is bounded by the limits of the world. The reward structure for the original single-objective Puddleworld task [13] is effectively a form of scalarisation with fixed weights for the two objectives of reaching the goal quickly and avoiding the puddles. On each step on which the goal is not reached, the agent receives a penalty of -1. An additional penalty applies if the agent is within a puddle, equal to 400 multiplied by the distance to the puddle's edge. To convert this problem to a multiobjective task, we present the two penalties as separate elements of a reward vector (omitting the multiplication by 400, as it is no longer relevant). To facilitate the evaluation of the Pareto front, it was necessary to make several alterations to the original problem specification. The noise added to the movement of the agent was omitted. The policies were based on a 20x20 discretisation of the state-space (although the actual position of the agent in the environment was still modeled as a continuous value). The goal was enlarged from its original triangular shape to fill the entire 0.05 unit square in the top-right corner of the world. With these alterations in place, and through the application of several manually identified constraints, it was possible to identify all non-dominated policies to construct the Pareto front shown in Fig 5. The overall shape of the front is convex, but a closer inspection of the front reveals a number of subtle local concavities and linearities, as shown in Fig 6.

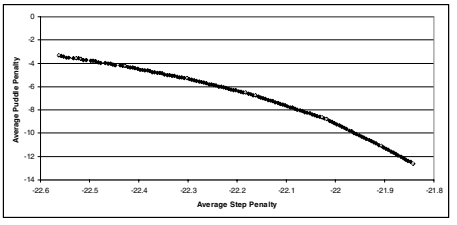


Fig. 5. The Pareto front for the MO-Puddle-world problem

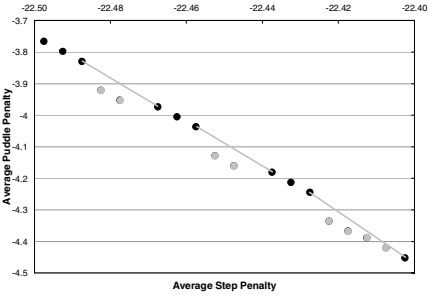
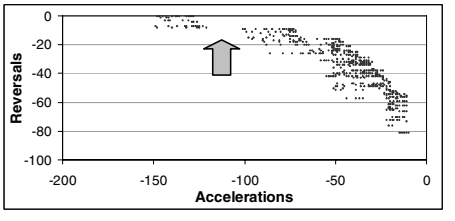
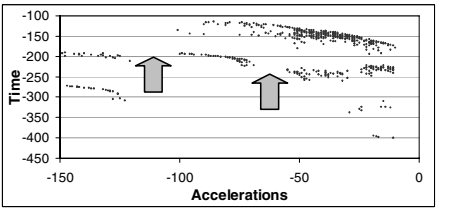
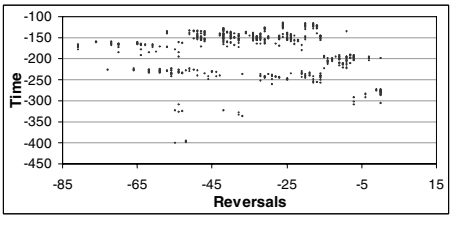


Fig. 6. A close-up view of Fig 5 (solutions in concave regions are shown in grey)







Figs. 7-9. 2-dimensional projections of the Pareto front for the Mountain-Car task, relative to each pair of objectives. The arrows indicate some of the discontinuities in this front.

MO-Mountain-Car. A car must escape from a 1-dimensional valley. The engine is less powerful than gravity, and so the car must reverse up the left side of the valley to build enough momentum to escape from the right side. The inputs are the current position and velocity, and there are 3 actions – accelerate, reverse, and zero throttle. In the single-objective case a penalty of -1 is received on all steps on which the goal-state is not reached [14]. To test the generality of MORL systems, some benchmarks should involve more than two objectives so two further objectives were added – minimising both the number of reversing and the number of acceleration actions. A penalty of -1 is received in the corresponding reward vector element when one of these actions is executed. Interestingly in the single-objective case the zero throttle action is redundant, whereas in the multiobjective task, the choice of when to choose zero throttle is a key difference between policies. As with MO-Puddleworld, it was necessary to restrict the policies to a discretised state space in order to evaluate the front – in this case a 6x6 discretisation was used. The resulting front is shown in

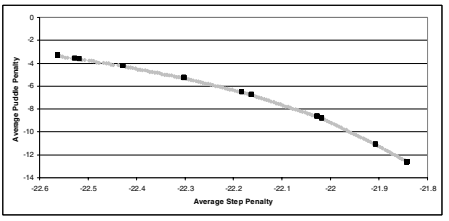


Fig. 10. The Pareto front of the MO-Puddle-world task (in grey) with the policies found by scalarisation superimposed (in black)

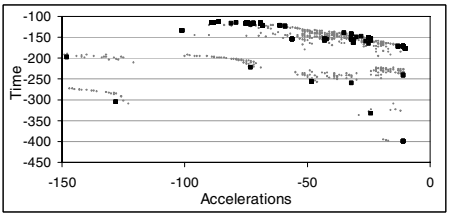


Fig. 11. The Pareto front of the MO-Mountain-Car task (in grey) relative to the acceleration and time objectives, with the policies found by scalarisation superimposed (in black)

Figs 7-9 – as it is difficult to interpret a 3-dimensional view, 2-dimensional projections have been provided. It should be noted that this front only considers policies which actually escape from the valley – this would need to be handled as a constraint by any MORL system, as it does not directly arise from the reward structure.

The performance of scalarisation on the benchmarks was assessed by applying scalarising weights to each known solution, and determining which returned the highest scalarised value. This was repeated for a variety of weights, and the resulting sets of solutions compared to the actual Pareto front. As noted earlier Deep Sea Treasure has a globally concave Pareto front. Whatever weight combinations are applied to this task, only the two extremal points will ever return the maximal scalarised value - the intermediate solutions will be overlooked by any system based on scalarisation. The limitations of scalarisation on MO-Puddleworld are more subtle, but nonetheless substantial. Searching the weight space in steps of 0.01 results in 101 weight sets being applied to the solutions, yet finds only 11 of the 141 unique policies present in the original Pareto front. Fig 10 compares these policies to the true front. It can be seen that the sparse set of policies found by scalarisation is a poor representation of the near continuous true front. A similar result is observed for MO-Mountain-Car. The need to assign weights for all 3 objectives increases the size of the weight-space to be searched, so using a step-size of 0.1 results in 66 weight combinations yet only detects 28 of the 470 unique solutions in the actual front. A step-size of 0.05 tests a 231 weight combinations, and locates only 41 policies. Fig 11 compares these 41 policies against the actual Pareto front. Once again, the approximate front produced by scalarisation is a poor match for the actual front. In particular the distribution is quite different from that of the actual front.

4 Conclusions and Future Work

This paper has argued for following the lead of MOO research, by developing MORL systems which approximate the Pareto front of policies, rather than producing just a single solution. Presenting the user with a range of solutions provides more information about the trade-offs between objectives, and allows an informed decision without the need to impose *a priori* biases on the nature of the solution. We have examined the use of scalarisation within Pareto-based MORL showing that it may be unable to produce a good approximation of the Pareto front for many problems, due to its inability to locate policies in non-convex regions of the front. Whilst the limitations of scalarisation are

well known amongst MOO researchers, they have been overlooked by many MORL practitioners. Hence we believe that there is a pressing need for MORL algorithms designed expressly to address the issue of deriving an approximation to the Pareto front. [11] has provided pioneering work in this area. It may also prove fruitful to extend existing methods such as [5], by performing multiple iterations of the algorithm using different values for the thresholds.

This paper also presented 3 test problems which are the first MORL tasks with identified Pareto fronts, making them valuable benchmarks for future research. To support such further use, it is important that this set of benchmarks be extended as the problems presented share a number of features such as their limited state-space dimensionality, non-stochasticity and episodic nature. Whilst these features aided in finding the Pareto fronts, they also simplify the task facing a learning agent. Therefore these benchmarks need to be augmented with a more diverse range of problems to better represent the challenges posed to MORL systems in real-world tasks.

References

1. Coello Coello, C.A.: Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. In: 2000 Congress on Evolutionary Computation, vol. 1, pp. 30–37 (2000)
2. Tesauro, G., Das, R., Chan, H., Kephart, J.O., Lefurgy, C., Levine, D.W., Rawson, F.: Managing power consumption and performance of computing systems using reinforcement learning. Neural Information Processing Systems (2007)
3. Natarajan, S., Tadepalli, P.: Dynamic preferences in multi-criteria reinforcement learning. In: 22nd International Conference on Machine Learning, Bonn, Germany, pp. 601–608 (2005)
4. Castelletti, A., Corani, G., Rizzolli, A., Soncinie-Sessa, R., Weber, E.: Reinforcement learning in the operational management of a water system. In: IFAC Workshop on Modeling and Control in Environmental Issues, Keio University, Yokohama, Japan, pp. 325–330 (2002)
5. Gabor, Z., Kalmar, Z., Szepesvari, C.: Multi-criteria reinforcement learning. In: The Fifteenth International Conference on Machine Learning, pp. 197–205 (1998)
6. Geibel, P.: Reinforcement learning with bounded risk. In: Proceedings of the 18th International Conference on Machine Learning (2001)
7. Geibel, P., Wysotzki, F.: Risk-sensitive reinforcement learning applied to control under constraints. Journal of Artificial Intelligence Research 24, 81–108 (2005)
8. Mannor, S., Shimkin, N.: The steering approach for multi-criteria reinforcement learning. In: Neural Information Processing Systems, Vancouver, Canada, pp. 1563–1570 (2001)
9. Mannor, S., Shimkin, N.: A geometric approach to multi-criterion reinforcement learning. Journal of Machine Learning Research 5, 325–360 (2004)
10. Shelton, C.R.: Importance sampling for reinforcement learning with multiple objectives, Massachusetts Institute of Technology AI Lab Technical Report No. 2001-003 (2001)
11. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: PESA-II: region-based selection in evolutionary multiobjective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 283–290 (2001)
12. Coello Coello, C.A., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithm MOP Approaches (Chapter Two). In: Evolutionary Algorithms for Solving Multiobjective Problems. Kluwer Academic Publishers, Dordrecht (2002)
13. Boyan, J.A., Moore, A.W.: Generalization in reinforcement learning: Safely approximating the value function. In: Neural Information Processing Systems (1995)
14. Sutton, R.S.: Generalisation in reinforcement learning: Successful examples using sparse coarse coding. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems, pp. 1038–1044 (1996)

An Approach for Generalising Symbolic Knowledge

Richard Dazeley[1] and Byeong-Ho Kang[2]

[1] School of Information Technology and Mathematical Sciences,
University of Ballarat, Ballarat, Victoria 7353, Australia
[2] School of Computing and Information Systems,
University of Tasmania, Hobart, Tasmania, 7001
r.dazeley@ballarat.edu.au, bhkang@utas.edu.au

Abstract. Many researchers and developers of knowledge based systems (KBS) have been incorporating the notion of context. However, they generally treat context as a static entity, neglecting many connectionists' work in learning hidden and dynamic contexts, which aids generalization. This paper presents a method that models hidden context within a symbolic domain achieving a level of generalisation. Results indicate that the method can learn the information that experts have difficulty providing by generalising the captured knowledge.

Keywords. hidden context, knowledge based systems, knowledge representation, ripple-down rules, situation cognition.

1 Introduction

The situation cognition (SC) view of knowledge revolves around the premise that knowledge is generated at the time of its use. This implies that the existence of knowledge is based on the context of a given situation [1, 2]. Methodologies, such as Formal Concept Analysis (FCA) [3], Repertory Grids [4] and Ripple-Down Rules (RDR) [5], have adopted a weak SC position by including contextual information. These approaches either incorporated the context directly in the knowledge or in the representation structure. However, they assume that the context is *a priori*, and therefore, deductive. This assumption leads to static representations, however, context in certain situations could be considered *a posteriori*, and therefore, inductive [6].

This paper presents a method that moves away from these contextually static representations and instead is designed to handle an Intermediate SC [7] view by including hidden and dynamic contexts. The results in this paper illustrate the method's ability to learn quickly while maintaining the ability to generalise. This notion of a generalising symbolic based system capable of finding hidden contextual information, led to the notion of combining a Knowledge Based System (KBS) with an Artificial Neural Network (ANN). The KBS selected was Multiple Classification RDR (MCRDR), as this is the most capable of modelling multiple contexts [8].

2 Multiple Classification Ripple-Down Rules (MCRDR)

Ripple-Down Rules is a maintenance centred methodology for a KBS based approach using the concept of fault patching [9] and was first proposed by [5]. It utilises a binary

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 379–385, 2008.

tree as a simple exception structure aimed at partially capturing the context that knowledge is obtained from an expert. The context is the sequence of rules that had evaluated to provide the given conclusion [5, 10-14]. Therefore, if the expert disagrees with a conclusion made by the system they can change it by adding a new rule. The new rule will only fire if the same path of rules is evaluated [12].

Ripple-Down Rules has been shown to be a highly effective tool for knowledge acquisition (KA) and knowledge maintenance (KM). However, it lacks the ability to handle tasks with multiple possible conclusions. Multiple Classification Ripple-Down Rules (MCRDR) aim was to redevelop the RDR methodology to provide a general method of building and maintaining a knowledge base (KB) for multiple classification domains. The methodology developed by [15] is based on the proposed solution by [11, 12]. The primary shift was to switch from the binary tree to an *n-ary* tree representation. Knowledge is acquired by inserting new rules into the MCRDR tree when a misclassification has occurred. The new rule must allow for the incorrectly classified case, identified by the expert, to be distinguished from the existing stored cases that could reach the new rule [16]. This is accomplished by the user identifying key differences between the current case and each of the rules' cornerstone cases.

3 Rated MCRDR

The hybrid methodology used in this paper, referred to as Rated MCRDR (RM), combines MCRDR with an artificial neural network (ANN). This function fitting algorithm learns patterns of fired rules found during the inferencing process. Firstly, a case is presented to the MCRDR tree, which classifies the case. Then for each rule in the inference, an associated input neuron will fire. The network then produces a vector of output values, $\overline{v}$, for the case presented. The system, therefore, provides two separate outputs; the case's classifications and an associated set of values.

Learning in RM is achieved in two ways. Firstly, the value for each corresponding value for $\overline{v}$ receives feedback from the environment concerning its accuracy. The network learns by either using the standard backpropagation approach using a sigmoid thresholding function, and the MCRDR component still acquires knowledge in the usual way. The only exception is when the expert adds a new rule to MCRDR. As the input space grows, new input nodes need to be added to the network in such a way that does not damage already learned information. Therefore, the network structure needed to be altered by adding shortcut connections from any newly created input nodes directly to each output node and using these connections to carry a weight adjustment. When a new input node is added, additional hidden nodes are added.

The *single-step-Δ-initialisation-rule,* Equation 1, directly calculates the required weight for the network to step to the correct solution immediately. This is accomplished by reversing the feedforward process back through the inverse of the symmetric sigmoid. It is possible for the expert to add multiple new rules for the one case. In these situations the calculated weight is divided by the number of new features, *m*. Finally, the equation is multiplied by the step-distance modifier, *Zeta (ζ)*. *Zeta (ζ)* should always be set in the range $0 \leq \zeta \leq 1$. It allows adjustments to how large a step should be taken for the new features.

$$w_{no} = \zeta \left[\left(\left(\log \left[\frac{f(net)_o + \delta_o + 0.5}{0.5 - (f(net)_o + \delta_o)} \right] \Big/ k \right) - \left[\left(\sum_{i=0}^{n-1} x_i w_{io} \right) + \left(\sum_{h=0}^{q} x_h w_{ho} \right) \right] \right) \Big/ mx_n \right] \quad (1)$$

4 Experiments and Results

The following results illustrate how RM compares against the two underlying methodologies: MCRDR and a backpropagation neural network. The output from RM's ANN consists of a vector, $\bar{v}$, of outputs. Each output will relate to one possible classification. There is an output for each of the class types in the dataset being tested. If the output is positive then it will be regarded as providing that classification. The same output method is used for the backpropagation method being compared against. The results presented in this paper use 10 different randomisations of the relevant dataset. The test investigates how the methods can correctly classify cases. In this test each dataset is divided into ten equal sized epochs. Results are presented where $9/10^{ths}$ of the dataset are used for training and $1/10^{th}$ for testing. The size of the training set is then reduced incrementally in steps of $1/10^{th}$, down to $1/10^{th}$.

4.1 Simulated Expertise and Datasets

Three Simulated experts were used in this study. The first simulated expert uses a C4.5 [17] decision tree to select symbols If the KB being constructed, incorrectly classifies a case then the simulated expert's decision tree is used to find attributes within rules that led to the correct classification. The problem with this simulated expert is that it requires an induction system, which is limited to single classification. However, the system in this paper is primarily targeting multiple classification domains. Thus, a second simulated expert was created that calculates its classifications based on each case's attributes. It uses a randomly generated table of values representing the level each attribute contributes to each class.

A third simulated expert was designed to provide a non-linearity therefore the classification needs to vary according to combinations of attributes. This was achieved by randomly pairing attributes together for each class. Once paired, they were given an increasing absolute value with an alternate sign. When a case is presented to the expert it is tested to see which class it belongs to by adding all the associated values and attribute pairs in each class. The expert will then classify the case according to which classes provided a positive, > 0, total.

RM and the underlying methods were compared across six datasets: chess, tic-tac-toe (TTT), nursery, audiology, car evaluation and a generated multi-class dataset. The first five, from [18], were tested using the *C4.5* simulated expert. The sixth was used twice, tested the methods, using the two multi-class simulated experts.

4.2 Results

The ability of the RM and the two base methods to generalise is measured by how well they can classify cases previously never seen. The performance of each method

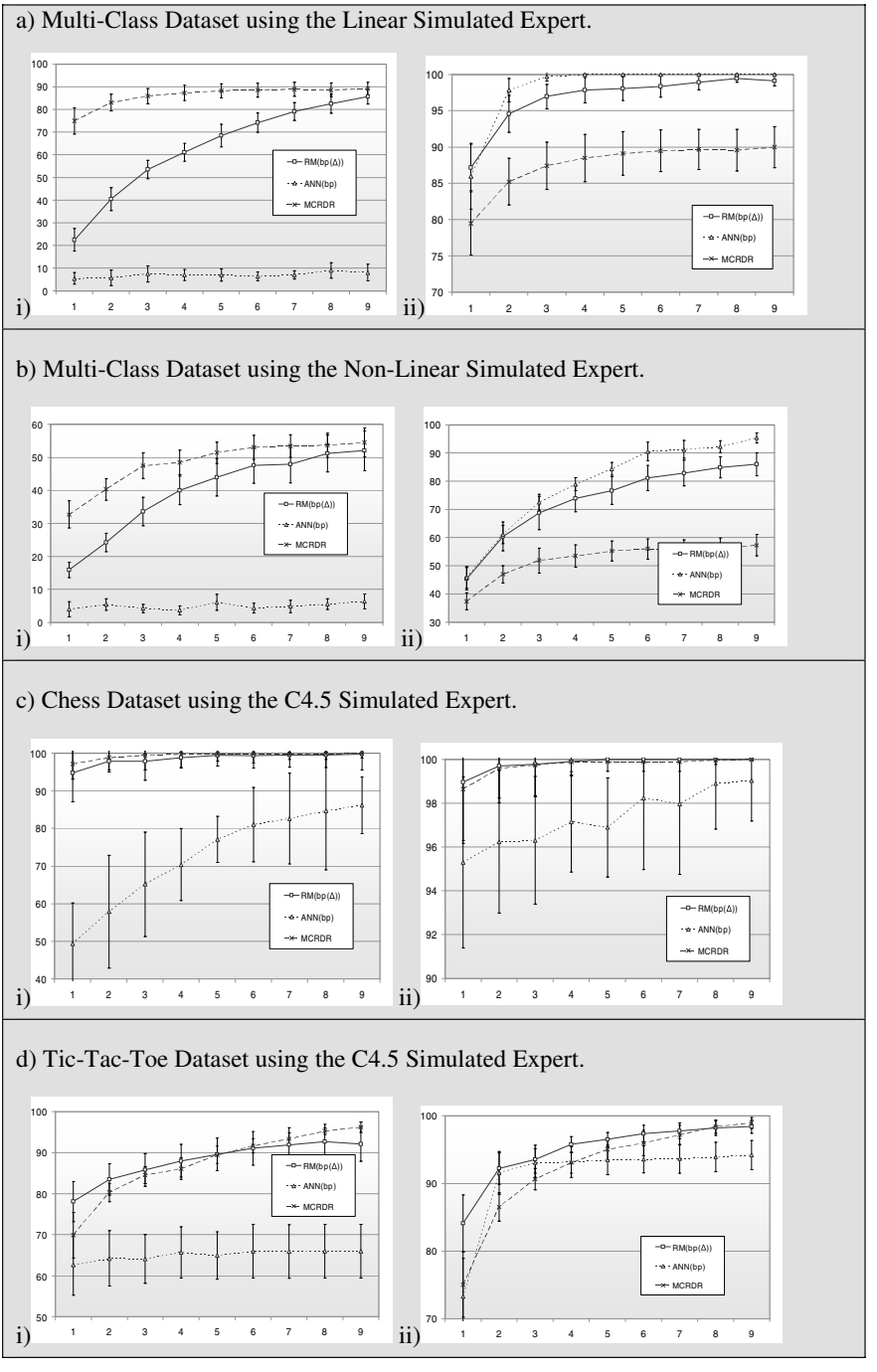


Fig. 1. (a-g) show two charts comparing the performance of RM, an ANN and MCRDR on 6 datasets. i) after one viewing of the training set. ii) after training was completed.

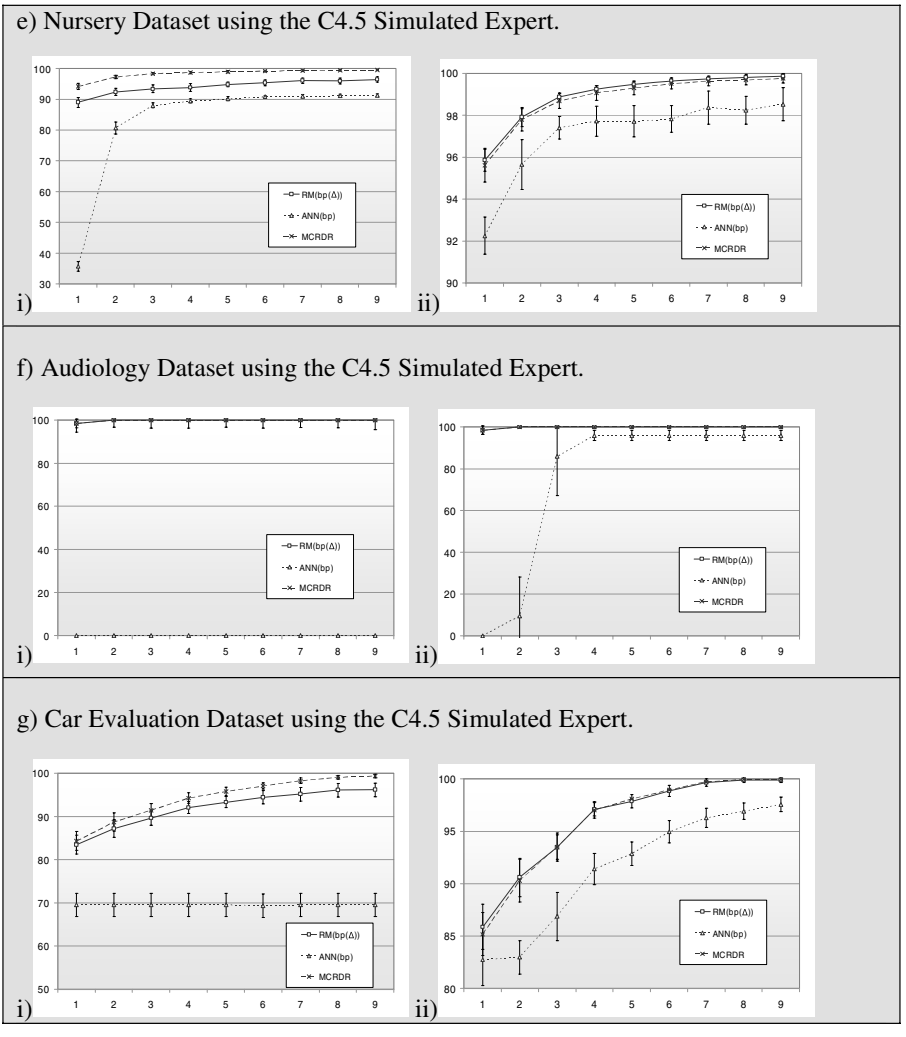


Fig. 1. *(continued)*

These results show that the RM hybrid system has done exceptionally well both initially as well as after training is complete. In both of the multi-class experiments RM significantly outperformed the neural network after the first training round. However, while it performed well, it was not able to match the performance of MCRDR. Also, after training was completed RM significantly generalised better than MCRDR but could not match the neural network. Although, it should be noted, that this was rarely a statistical significant difference in performance.

RM's performance on the chess, TTT, Audiology and Car Evaluation datasets (and to a lesser extent the Nursery dataset) was considerably better than in the multi-class environments. For instance, it was able to match MCRDR in the first iteration on a

number of the datasets. Additionally, it was able to learn a much improved generalisation function over both the neural network and to a lesser extent MCRDR after training.

While RM was not always able to perform quite as well as hoped, some points should be noted. For instance, when it did not reach its full potential it did come close. Secondly, in the multi-class datasets, it was also found that after the second training iteration RM had outperformed MCRDR across most of the seven tests. It should also be noted that the results for the neural network had required significantly more training to get its marginally better results.

5 Conclusion

This paper presented an algorithm that detects and models hidden contexts within a symbolic domain. The method developed builds on the already established Multiple Classification Ripple-Down Rules (MCRDR) approach and was referred to as Rated MCRDR (RM). RM retains a symbolic core that acts as a contextually static memory, while using a connection based approach to learn a deeper understanding of the knowledge captured.

A number of results were presented, which have shown how RM is able to acquire knowledge and learning. RM's ability to perform well can be put down to two features of the system. First, is that the flattening out of the dimensionality of the problem domain by the MCRDR component allows the system to learn a problem that is mostly linear even if the original problem domain was non-linear. This allows the network component to learn significantly faster. Second, the network gets an additional boost through the *single-step-Δ-initialisation rule*, allowing the network to start closer to the correct solution when knowledge is added.

Acknowledgements

The majority of this paper is based on research carried out while affiliated with the Smart Internet Technology Cooperative Research Centre (SITCRC) Bay 8, Suite 9/G12 Australian Technology Park Eveleigh NSW 1430 and the School of Computing, University of Tasmania, Locked Bag 100, Hobart, Tasmania.

References

1. Menzies, T.: Assessing Responses to Situated Cognition. In: Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, Catelonia, Spain (1996)
2. Menzies, T.: Towards Situated Knowledge Acquisition. International Journal of Human-Computer Studies 49, 867–893 (1998)
3. Wille, R.: Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. In: Rival, I. (ed.) Ordered Sets: Proceedings of the NATO Advanced Study Institute held at Banff, Canada, pp. 445–472. D. Reidel Publishing (1981)
4. Kelly, G.A.: The Psychology of Personal Constructs, Norton, New York (1955)

5. Compton, P., Jansen, R.: Knowledge in Context: a strategy for expert system maintenance. In: Second Australian Joint Artificial Intelligence Conference (AI 1988), vol. 1, pp. 292–306 (1988)
6. Brezillon, P.: Context in Artificial Intelligence: II. Key elements of contexts. Computer and Artificial Intelligence 18(5), 425–446 (1999)
7. Dazeley, R., Kang, B.: Epistemological Approach to the Process of Practice. Journal of Mind and Machine (in press, 2008)
8. Gaines, B.: Knowledge Science and Technology: Operationalizing the Enlightenment. In: Proceedings of the 6th Pacific Knowledge Acquisition Workshop, Sydney, Australia, pp. 97–124 (2000)
9. Menzies, T., Debenham, J.: Expert System Maintenance. In: Kent, A., Williams, J.G. (eds.) Encyclopaedia of Computer Science and Technology, vol. 42, pp. 35–54. Marcell Dekker, New York (2000)
10. Beydoun, G.: Incremental Acquisition of Search Control Heuristics, PhD thesis (2000)
11. Compton, P., Edwards, G., Kang, B.: Ripple Down Rules: Possibilities and Limitations. In: 6th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW 1991), vol. 1, pp. 6.1–6.18. SRDG publications, Canada (1991)
12. Compton, P., Kang, B., Preston, P.: Knowledge Acquisition Without Knowledge Analysis. In: Aussenac, N., Boy, G.A., Ganascia, J.-G., Kodratoff, Y., Linster, M., Gaines, B.R. (eds.) EKAW 1993. LNCS, vol. 723, pp. 277–299. Springer, Heidelberg (1993)
13. Preston, P., Edwards, G., Compton, P.: A 1600 Rule Expert System Without Knowledge Engineers. In: Moving Towards Expert Systems Globally in the 21st Century (Proceedings of the Second World Congress on Expert Systems 1993), New York, pp. 220–228 (1993)
14. Preston, P., Edwards, G., Compton, P.: A 2000 Rule Expert System Without a Knowledge Engineer. In: Proceedings of the 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, pp. 17.1–17.10 (1994)
15. Kang, B.: Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules, PhD thesis (1996)
16. Kang, B.H., Compton, P., Preston, P.: Multiple Classification Ripple Down Rules: Evaluation and Possibilities. In: The 9th Knowledge Acquisition for Knowledge Based Systems Workshop, SRDG Publications, Department of Computer Science, University of Calgary, Banff, Canada (1995)
17. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
18. Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences (1998)

Single-Cycle Image Recognition Using an Adaptive Granularity Associative Memory Network

Anang Hudaya Muhamad Amin and Asad I. Khan

Clayton School of Information Technology,
Monash University, Clayton, 3168 VIC, Australia
{Anang.Hudaya,Asad.Khan}@infotech.monash.edu.au

Abstract. Pattern recognition involving large-scale associative memory applications, generally constitutes tightly coupled algorithms and requires substantial computational resources. Thus these schemes do not work well on large coarse grained systems such as computational grids and are invariably unsuited for fine grained environments such as wireless sensor networks (WSN). Distributed Hierarchical Graph Neuron (DHGN) is a single-cycle pattern recognising algorithm, which can be implemented from coarse to fine grained computational networks. In this paper we describe a two-level enhancement to DHGN, which enables it to act as a standard binary image recogniser. This paper demonstrates that our single-cycle learning approach can be successfully applied to denser patterns, such as black and white images. Additionally we are able to load-balance the pattern recognition processes, irrespective of the granularity of the underlying computational network.

Keywords: Associative Memory, Pattern Recognition, Distributed Hierarchical graph Neuron (DHGN), Complexity Estimation.

1 Introduction

The computational complexity of contemporary associative memory (AM) based pattern recognition algorithms is considerably high and difficult to be deployed in resource-constraint environment, such as mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs). The AM approaches such as Morphological Associative Memory (MAM) [1] and Fuzzy Associative Memory (FAM) [2] generally tend to be computationally intensive and iterative. In this paper, we present key enhancements to our single-cycle learning auto-associative memory scheme known as Distributed Hierarchical Graph Neuron (DHGN) [3,4]. These enhancements allow the scheme to be used for dense patterns, such as black and white bitmap images. DHGN is a single-cycle pattern recognition algorithm, which is based on Graph Neuron (GN) concept introduced by Khan [5] for single-cycle pattern recognition. DHGN comprises sets of replicated lightweight GN processes. These sets, termed as DHGN compositions [3], can be readily distributed from relatively coarse systems such as computational grids

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 386–392, 2008.

to finely distributed resource-constraint networks for a variety of applications. With regards to the distorted/noisy patterns, DHGN offers comparable accuracy to conventional algorithms due to the work done by Nasution and Khan on Hierarchical Graph Neuron (HGN) [6]. DHGN algorithm could be efficiently deployable from coarse grained to fine grained networks. Our single-cycle learning approach, though being highly scalable with respect to network granularity and able to store a very large number of patterns in single-cycles, has not so far been tested for dense patterns. The analyses in this paper will show that our single-cycle learning scheme works well with binary images containing noise/distortions and missing information.

The paper is divided into 6 sections. Section 2 provides a brief overview of DHGN algorithm. Section 3 estimates the complexity of the recognition process within DHGN algorithm in terms of its bias array storage capacity. Section 4 describes the voting mechanism adopted as an enhancement to the original DHGN algorithm for the overall pattern-level recognition. Section 5 explains the results of the binary image pattern recognition tests. Section 6 concludes the paper.

2 DHGN Algorithm Overview

DHGN algorithm applies the adjacency comparison in its recognition approach. In this regard the input patterns are automatically synthesised into subpatterns by the GN processes. The sets of GN processes concurrently compare their subpatterns with the historical information, which is locally available within each GN. Finally these processes produce their independent recall/memorise outputs. These outputs are integrated by the network to effect an overall recall/store operation. Details of the DHGN implementation can be found in [3,4,7].

DHGN network is composed of a number of DHGN subnets and a Stimulator/Interpreter Module (SI Module) node. Fig. 1 shows the decomposition of binary image pattern 'K' into subpatterns and each of the subpatterns being fed into the DHGN network. The SI input activates the GN nodes corresponding to the bits of the input pattern. In doing so each subpattern is automatically decomposed and mapped to relevant GNs in the subnets. Each subnet integrates its responses and sends the results to the SI Module to form an overall response.

Each DHGN subnet is composed of a number of processing nodes, depending on the size of the subpatterns sp_{size} and the number of possible input values within the pattern, p_{input}. Equation (1) shows the number of processing (GN) nodes n_{GN} within a single DHGN subnet.

$$n_{GN} = p_{input} \left(\frac{sp_{size} + 1}{2} \right)^2 . \tag{1}$$

The adjacency information within a single processing node is kept in a data structure known as bias array. Each new message received from adjacent nodes is treated as a bias entry with its own bias index. The recognition processes within DHGN heavily rely on this structure, where the status of the adjacent

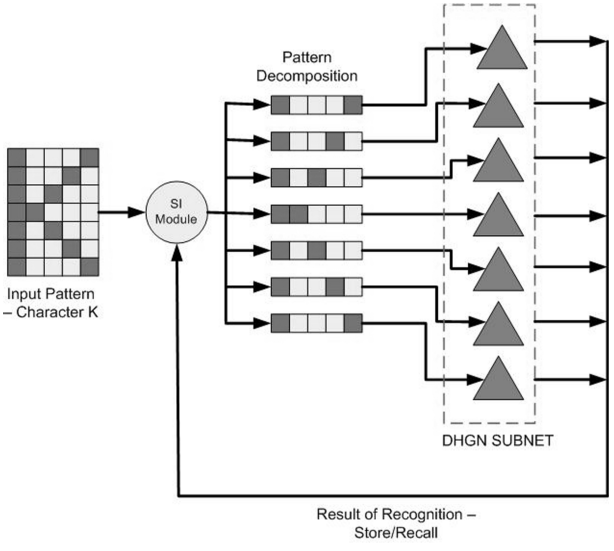


Fig. 1. Pattern recognition processes using DHGN algorithm where a 7x5 bitmap of letter K is mapped as subpatterns over 7 hierarchically formed GN sub-networks

preceding and succeeding columns' nodes is checked and compared for every new input using a match-compare approach. The bias array complexity estimate is included in the following section.

3 DHGN Bias Array Complexity

In this DHGN implementation we have adopted a two-level approach. The first level involves the recognition process at subpattern level and the second level is the recognition at the overall pattern level. This section will discuss the recognition at subpattern level, while the next section will outline the recognition at pattern level.

In a DHGN implementation, the recognition at subpattern level deals with the recognition process occurring within individual DHGN subnet. In this context, the complexity of the algorithm could be measured in terms of the size of the bias array as an indicator of the scalability of the algorithm. The size of the bias array determines the number of input subpatterns that can be stored within a single DHGN subnet. From the scalability point of view, this storage requirement must not significantly increase with an increase in the number of stored patterns [6].

The following equations show the bias array size estimation for DHGN implementation. These equations have been derived based on binary pattern recognition implementation. In this work, we have considered a one-dimensional implementation of DHGN algorithm, where a two dimensional pattern is input as a long bit string.

Base Level. For each non-edge processing node the maximum number of bias array entries is equivalent to the square values of the number of rows within each layer, n_r:

$$bs^n_{l_0} = n_r^2 \ . \tag{2}$$

For each processing node at the edge:

$$bs^e_{l_0} = n_r \ . \tag{3}$$

Therefore, the cumulative maximum size of bias array at the base level in each DHGN subnet could be derived as shown in (4).

$$bs^{max}_{l_0} = n_r^2 \, (sp_{size} - 2) + 2bs^e_{l_0} \ . \tag{4}$$

It may be noted from these equations that the size of the bias array at the base level is not affected by the number of subpatterns stored. This is due to the fact that the pattern storage occurs on the basis of adjacency information received from preceding and succeeding nodes within the subnet. Thus the maximum possible size for the bias arrays solely depends on the range of possible input values and size of the input subpatterns.

Middle Levels. The maximum size of the bias array at any middle level l_i is correlated with the maximum size of the bias array at the level below it. For non-edge processing node at middle level l_i, the maximum size of its bias array could be derived as follows:

$$bs^n_{l_i} = bs^n_{l_i - 1} * n_r^2 \ . \tag{5}$$

For each processing node at the edge, the following equation shows the maximum size of its bias array:

$$bs^e_{l_i} = bs^n_{l_i - 1} * n_r \ . \tag{6}$$

Therefore, the cumulative maximum size of bias array at the middle level within a DHGN subnet could be estimated with the following equation:

$$bs^{max}_{l_i} = bs^n_{l_i} \, (sp_{size} - (2i + 2)) + 2bs^e_{l_i} \ . \tag{7}$$

Top Level. At the top level processing node, the maximum size of the bias array could be derived from the previous level non-edge processing node's maximum bias array size.

From these equations, the total maximum size of all the bias arrays within a single DHGN subnet could be deduced as shown in (8):

$$bs^{max}_{total} = bs^{max}_{l_0} + \sum_{l_i=1}^{l_{top}-1} bs^{max}_{l_i} + bs^n_{l_{top}-1} \ . \tag{8}$$

The DHGN bias array has been specifically designed to store the adjacency information, as compared to the actual pattern data. Fig. 2 shows the storage mechanism employed by DHGN algorithm for two different patterns.

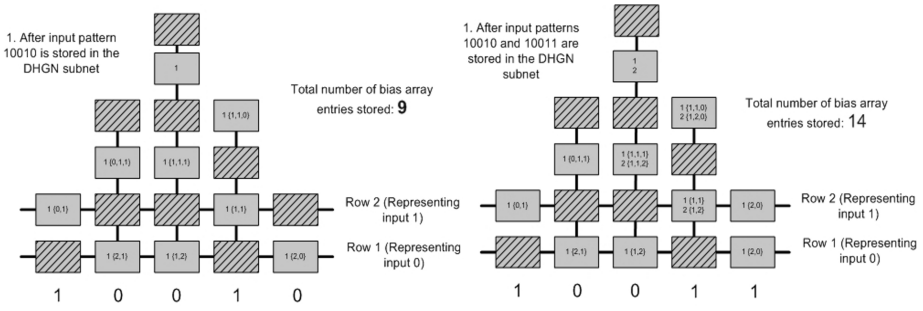


Fig. 2. Pattern adjacency storage mechanism employed by DHGN. Two different patterns were stored in this DHGN subnet, increasing the total number of bias array entries within the subnet from 9 to 14.

From the figure it may be noted that the second pattern input only contribute about 35% increment in the number of bias array entries stored. This increment is far less than the estimated increment of storage requirement if both actual patterns are stored without their adjacency information.

4 Pattern-Level Recognition

Recognition at pattern level in DHGN implementation involves the voting mechanism, adopted from the work by Cruz et al [8]. Some modifications have been carried out to suit the mechanism for our DHGN approach.

Within this scheme, it is a requirement for SI module to be able to store both the list of indices and the voting vectors. In addition, SI module should be capable of handling multiple simultaneous inputs from different DHGN subnets. Otherwise, the implementation would become a bottleneck at this stage.

Given P_1, P_2, and P_3 as binary patterns representing character A, E, and O respectively, and P_1^d represents the distorted pattern of P_1:

$$P_1 = \begin{pmatrix} 0\,1\,0 \\ 1\,0\,1 \\ 1\,0\,1 \\ 1\,1\,1 \\ 1\,0\,1 \end{pmatrix} \quad P_2 = \begin{pmatrix} 1\,1\,1 \\ 1\,0\,0 \\ 1\,1\,0 \\ 1\,0\,0 \\ 1\,1\,1 \end{pmatrix} \quad P_3 = \begin{pmatrix} 1\,1\,1 \\ 1\,0\,1 \\ 1\,0\,1 \\ 1\,0\,1 \\ 1\,1\,1 \end{pmatrix} \quad P_1^d = \begin{pmatrix} 0\,1\,1 \\ 1\,0\,1 \\ 1\,0\,1 \\ 1\,1\,1 \\ 1\,0\,1 \end{pmatrix} .$$

These patterns are decomposed into 5 subpatterns, where each subpattern represents a row of the overall character pattern. Hence, the S vector containing the indices retrieved from the DHGN subnets (for these non-distorted patterns) would have the following entries:

$$S = \begin{pmatrix} 1\,1\,1\,1\,1 \\ 2\,2\,2\,2\,2 \\ 2\,1\,1\,3\,2 \end{pmatrix} .$$

Note that the index of S_{12} is similar to S_{32}, due to the fact that it is a recalled subpattern of pattern P_1 at subpattern 2. With the introduction of the distorted pattern into the network, the indices retrieved from DHGN subnet will be recorded as a vector R:

$$R = \begin{pmatrix} 4 & 1 & 1 & 1 & 1 \end{pmatrix} \quad .$$

This simply shows that at subpattern 1, a new subpattern has been introduced and the rest of the subpatterns are recalled as subpatterns from either subpatterns of P_1 or P_3. Therefore, the vector W would have the following entries:

$$W = \begin{pmatrix} 4 & 0 & 0 \end{pmatrix} \quad .$$

This shows that there are 4 occurrences of subpatterns P_1 and 0 occurrences of either subpattern P_2 or P_3. The maximum entry is at W_1 which corresponds to pattern P_1. Therefore pattern P_1 will be recalled.

5 Results and Discussion

For the purpose of ensuring the accuracy of our proposed two-level DHGN algorithm for pattern recognition, we have conducted a series of tests involving binary patterns. These binary patterns are grey-scale images which have been dimensionally reduced to binary values. We have considered a Gaussian noise distortion and block distortion tests. In doing this, we have used 20 heterogeneous binary images with the size of 128 x 128 bits.

The results of the recognition tests have shown that DHGN algorithm exerts high recall accuracy for patterns with Gaussian noise and block-distortion. Furthermore, its recall accuracy does not get affected with an increase in the number of patterns being stored. However, it should be noted that the images that have been used in these tests are heterogeneous images, i.e. non-similar images. Fig. 3 shows the results of these tests.

Fig. 3. The recognition of image Lena from Gaussian-distorted images and block-distorted images. DHGN produces perfect recall for each of the distorted images presented.

6 Conclusions

The proposed two-Level DHGN algorithm, for binary pattern, works at the sub-pattern level and later at the pattern level. This enhanced DHGN provides high recall accuracy for binary images with distortions whilst retaining the single-cycle characteristic of the GN approach. The results of the analysis also prove that the bias array complexity in DHGN algorithm is not heavily affected by the increase in the number of stored patterns. This is due to the fact that DHGN only retains the adjacency information for each pattern through its bias array storage scheme. In addition, the proposed two-level DHGN approach for pattern recognition has been designed to suit both large coarse-grained network environment, as well as fine-grained network environment.

References

1. Ritter, G.X., Sussner, P., Diaz-de-Leon, J.L.: Morphological Associative Memories. IEEE Transactions on Neural Networks 9, 281–293 (1998)
2. Kosko, B.: Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence. Prentice-Hall, Englewood Cliffs (1992)
3. Khan, A.I., Muhamad Amin, A.H.: One Shot Associative Memory Method for Distorted Pattern Recognition. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 705–709. Springer, Heidelberg (2007)
4. Muhamad Amin, A.H., Khan, A.I.: Commodity-Grid Based Distributed Pattern Recognition Framework. In: Sixth Australasian Symposium on Grid Computing and e-Research (AusGrid 2008), Wollongong, NSW, Australia (2008)
5. Khan, A.I.: A Peer-to-Peer Associative Memory Network for Intelligent Information Systems. In: The Proceedings of The Thirteenth Australasian Conference on Information Systems, Melbourne, Australia (2002)
6. Nasution, B.B., Khan, A.I.: A Hierarchical Graph Neuron Scheme for Real-Time Pattern Recognition. IEEE Transactions on Neural Networks 19, 212–229 (2008)
7. Muhamad Amin, A.H., Mahmood, R.A.R., Khan, A.I.: Analysis of Pattern Recognition Algorithms using Associative Memory Approach: A Comparative Study between the Hopfield Network and Distributed Hierarchical Graph Neuron (DHGN). In: IEEE 8th International Conference on Computer and Information Technology (CIT 2008), Sydney, NSW, Australia (2008)
8. Cruz, B., Sossa, H., Barron, R.: A New Two-Level Associative Memory for Efficient Pattern Restoration. Neural Processing Letters (2006)

Combined Pattern Mining: From Learned Rules to Actionable Knowledge*

Yanchang Zhao[1], Huaifeng Zhang[1], Longbing Cao[1],
Chengqi Zhang[1], and Hans Bohlscheid[2]

[1] Data Sciences & Knowledge Discovery Research Lab
Centre for Quantum Computation and Intelligent Systems
Faculty of Engineering & IT, University of Technology, Sydney, Australia
{yczhao,hfzhang,lbcao,chengqi}@it.uts.edu.au
[2] Projects Section, Business Integrity Programs Branch,
Centrelink, Australia
hans.bohlscheid@centrelink.gov.au

Abstract. Association mining often produces large collections of association rules that are difficult to understand and put into action. In this paper, we have designed a novel notion of *combined patterns* to extract useful and actionable knowledge from a large amount of learned rules. We also present definitions of combined patterns, design novel metrics to measure their interestingness and analyze the redundancy in combined patterns. Experimental results on real-life social security data demonstrate the effectiveness and potential of the proposed approach in extracting actionable knowledge from complex data.

1 Introduction

The notion of association rules [1] was proposed 15 years ago and is widely used today. However, as large numbers of association rules are often produced by association mining, it can sometimes be very difficult for users to not only understand such rules, but also find them a useful source of knowledge to apply to their business processes. Therefore, to present associations in an interesting and effective way, and in order to find actionable knowledge from resultant association rules, a novel idea of *combined patterns* is proposed. Combined patterns comprise *combined association rules*, *combined rule pairs* and *combined rule clusters*. A combined association rule is composed of multiple heterogeneous itemsets from different datasets, while combined rule pairs and combined rule clusters are built from combined association rules. The proposed combined patterns provide more interesting knowledge and more actionable results than traditional association rules. The contributions of this paper are: 1) a definition of combined

* This work was supported by the Australian Research Council (ARC) Linkage Project LP0775041 and Discovery Projects DP0667060 & DP0773412, and Early Career Researcher Grant RM2007002447 from University of Technology, Sydney, Australia.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 393–403, 2008.

patterns, including combined rules, combined rule pairs and combined rule clusters; 2) interestingness measures designed for combined patterns; 3) two kinds of redundancy (i.e., rule redundancy and rule pair redundancy) identified for combined patterns; and 4) an experimental evaluation of the proposed technique on real-life data.

2 Related Work

There are often too many association rules discovered from a dataset and it is necessary to conduct post-processing before a user is able to study the rules and identify interesting ones from them. There are many techniques proposed to summarize and/or post-analyze the learned association rules [3,6]. Hilderman et al. proposed to characterize itemsets with information from external databases, e.g., customer or lifestyle data [2]. Their technique works by firstly mining frequent itemsets from transactional data and then partitioning each frequent itemset according to the corresponding characteristic tuple. This method likely results in a large number of rules when many characteristics are involved, with every characteristic having multiple value. Liu and Hsu proposed to rank learned rules by matching against expected patterns provided by user [4]. *Rule_Similarity* and *Rule_Difference* are defined to compare the difference between two rules based on their conditions and consequents, and *Set_Similarity* and *Set_Difference* are defined to measure the similarity between two sets of rules. The learned rules are ranked by the above similarity/difference and then it is up to the user to identify interesting patterns. In another work, Liu et al. proposed to mine for *class association rules* and build a classifier based on the rules [5]. With their rule generator, the rule with the highest confidence is chosen from all the rules having the same conditions but different consequents. Liu et al. also proposed *direction setting rules* to prune and summarize association rules [6]. Chi-square (χ^2) test is used to measure the significance of rules and insignificant ones are pruned. The test is then used again to remove the rules with "expected directions", that is, the rules which are combinations of direction setting rules. Zaïane and Antonie studied strategies for pruning classification rules to build associative classifiers [7]. Their idea selects rules with high accuracy based on the plot of correct/incorrect classification for each rule on the training set. Lent et al. proposed to reduce the number of learned association rules by clustering [3]. Using two-dimensional clustering, rules are clustered by merging numeric items to generate more general rules.

3 The Problem

The example that follows illustrates the target problem. Suppose that there are two datasets, transactional dataset and customer demographic dataset (see Tables 1 and 2), where "Churn" is the behaviour of a customer's switching from a company to another. In the following analysis, campaigns "d" and "e" are ignored to make the result easy to read. The traditional association rules

Table 1. Transactional Data

Customer ID	Campaign/Policy	Churn
1	a,b	Y
1	a	Y
2	a,c	N
2	b,c	Y
2	b,c,d	N
3	a,c,d	Y
3	a,b,e	Y
4	a,b	N
4	c	N
4	b,d	N

Table 2. Customer Demographic Data

Customer ID	Gender	...
1	F	
2	F	
3	M	
4	M	

Table 3. Traditional Association Rules

Rules	Supp	Conf	Lift
$F \rightarrow Y$	3/10	3/5	1.2
$F \rightarrow N$	2/10	2/5	0.8
$M \rightarrow Y$	2/10	2/5	0.8
$M \rightarrow N$	3/10	3/5	1.2
$a \rightarrow Y$	4/10	4/6	1.3
$a \rightarrow N$	2/10	2/6	0.7
$b \rightarrow Y$	3/10	3/6	1
$b \rightarrow N$	3/10	3/6	1
$c \rightarrow Y$	2/10	2/5	0.8
$c \rightarrow N$	3/10	3/5	1.2

Table 4. Combined Association Rules

Rules	Supp	Conf	Lift	$Lift_1$	$Lift_2$	I_{rule}
$F \wedge a \rightarrow Y$	2/10	2/3	1.3	1	1.1	0.8
$F \wedge b \rightarrow Y$	2/10	2/3	1.3	1.3	1.1	1.1
$F \wedge c \rightarrow N$	2/10	2/3	1.3	1.1	1.7	1.4
$M \wedge a \rightarrow Y$	2/10	2/3	1.3	1	1.7	1.3
$M \wedge b \rightarrow N$	2/10	2/3	1.3	1.3	1.1	1.1

Table 5. Combined Rule Pairs

Pairs	Combined Rules	I_{pair}
$\mathcal{P}_1$	$M \wedge a \rightarrow Y$ $M \wedge b \rightarrow N$	1.4
$\mathcal{P}_2$	$F \wedge b \rightarrow Y$ $M \wedge b \rightarrow N$	1.2

discovered are shown in Table 3, and the four rules with lift greater than one are $F \rightarrow Y$, $M \rightarrow N$, $a \rightarrow Y$ and $c \rightarrow N$. If partitioning the whole population into two groups, male and female, based on the demographic data in Table 2, and then mining the two groups separately, some rules are shown in Table 4, where $Lift_1$ and $Lift_2$ denote respectively the lift of the first/second part of the left side, and I_{rule} is the interestingness of the combined rule. The definitions of the three measures will be given in Section 4.2. We can see from Table 4 that more rules with high confidence and lift can be found by combining the rules from two separate datasets.

Although all the rules in Table 4 are of the same confidence and lift, their interestingness are not the same, which is shown by the last column I_{rule}. For example, for the first rule in Table 4, $F \wedge a \rightarrow Y$, its interestingness I_{rule} is 0.8, which indicates that the rule is not interesting at all. The explanation is that its lift is the same as the lift of $a \rightarrow Y$ (see Table 3), which means that

F contributes nothing in the rule. Therefore, our new measures are more useful than the traditional confidence and lift.

It is more interesting to organize the rules into contrasting pairs shown in Table 5, where I_{pair} is the interestingness of the rule pair. $\mathcal{P}_1$ is a rule pair for male group, and it shows that a is associated with churn but b with stay. $\mathcal{P}_1$ is actionable in that it suggests b is a preferred action/policy to keep male customers from churning. Moreover, male customers should be excluded when initiating campaign a. $\mathcal{P}_2$ is a rule pair with the same campaign but different demographics. With the same action b, male customers tend to stay, but female tend to churn. It suggests that b is a preferable action for male customers but an undesirable action for female customers.

From the previous example, we can see that rule pairs like $\mathcal{P}_1$ and $\mathcal{P}_2$ provide more information and are more useful and actionable than traditional simple rules shown in Table 3 and in this paper, they are referred to as *combined patterns*. A straightforward way to find the rules in Table 4 is to join Tables 1 and 2 in a pre-processing stage and then apply traditional association rule mining to the derived table. Unfortunately, it is often infeasible to do so in many applications where a dataset contains hundreds of thousands of records or more. Moreover, the rule clusters which organize related rules together are more useful and actionable than individual rules. To find the above useful knowledge like $\mathcal{P}_1$ and $\mathcal{P}_2$, a novel idea of combined patterns will be proposed in the next section.

4 Combined Pattern Mining

In this section we provide definitions of combined association rules and combined rule pairs/clusters, and then presents their interestingness and redundancy.

4.1 Definitions of Combined Patterns

Combined patterns take forms of *combined association rules*, *combined rule pairs* and *combined rule clusters*, which are defined as follows.

Definition 1 (Combined Association Rule). *Assume that there are k datasets $\mathcal{D}_i$ $(i = 1..k)$. Assume I_i to be the set of all items in datasets $\mathcal{D}_i$ and $\forall i \neq j,\ I_i \cap I_j = \emptyset$. A combined association rule R is in the form of*

$$A_1 \wedge A_2 \wedge ... \wedge A_k \rightarrow T, \tag{1}$$

where $A_i \subseteq I_i$ $(i = 1...k)$ is an itemset in dataset $\mathcal{D}_i$, $T \neq \emptyset$ is a target item or class and $\exists i, j, i \neq j, A_i \neq \emptyset, A_j \neq \emptyset$.

For example, A_1 can be a demographic itemset, A_2 can be a transactional itemset on marketing campaign, A_3 can be an itemset from a third-party dataset, and T can be the loyalty level of a customer. The combined association rules are then further organized into rule pairs by putting similar but contrasting rules together as follows.

Definition 2 (Combined Rule Pair). *Assume that R_1 and R_2 are two combined rules and that their left sides can be split into two parts, U and V, where U and V are respectively itemsets from $\mathcal{I}_U$ and $\mathcal{I}_V$ ($\mathcal{I} = \{I_i\}$, $\mathcal{I}_U \subset \mathcal{I}, \mathcal{I}_V \subset \mathcal{I}, \mathcal{I}_U \neq \emptyset, \mathcal{I}_V \neq \emptyset$ and $\mathcal{I}_U \cap \mathcal{I}_V = \emptyset$). If R_1 and R_2 share a same U but have different V and different right sides, then they build a combined rule pair $\mathcal{P}$ as*

$$\mathcal{P} : \begin{cases} R_1 : U \wedge V_1 \rightarrow T_1 \\ R_2 : U \wedge V_2 \rightarrow T_2 \end{cases}, \tag{2}$$

where $U \neq \emptyset$, $V_1 \neq \emptyset, V_2 \neq \emptyset, T_1 \neq \emptyset, T_2 \neq \emptyset, U \cap V_1 = \emptyset$, $U \cap V_2 = \emptyset$, $V_1 \cap V_2 = \emptyset$ and $T_1 \cap T_2 = \emptyset$.

A combined rule pair is composed of two contrasting rules, which suggests that for customers with same characteristics U, different policies/campaigns, V_1 and V_2, can result in different outcomes, T_1 and T_2. Based on a combined rule pair, related combined rules can be organized into a cluster to supplement more information to the rule pair.

Definition 3 (Combined Rule Cluster). *A combined rule cluster $\mathcal{C}$ is a set of combined association rules based on a combined rule pair $\mathcal{P}$, where the rules in $\mathcal{C}$ share a same U but have different V in the left side.*

$$\mathcal{C} : \begin{cases} U \wedge V_1 \rightarrow T_1 \\ U \wedge V_2 \rightarrow T_2 \\ \cdots \\ U \wedge V_n \rightarrow T_n \end{cases}, \tag{3}$$

where $U \neq \emptyset$; $\forall i, V_i \neq \emptyset, T_i \neq \emptyset, U \cap V_i = \emptyset$; and $\forall i \neq j, V_i \cap V_j = \emptyset$.

The rules in cluster $\mathcal{C}$ have the same U but different V, which makes them associated with various results T. Note that two rules in a cluster may have a same T. For example, assume that there is a rule pair $\mathcal{P}$ and a rule cluster $\mathcal{C}$ is built based on $\mathcal{P}$ by simply adding a third rule as follows.

$$\mathcal{P} : \begin{cases} U \wedge V_1 \rightarrow stay \\ U \wedge V_2 \rightarrow churn \end{cases}, \qquad \mathcal{C} : \begin{cases} U \wedge V_1 \rightarrow stay \\ U \wedge V_2 \rightarrow churn \\ U \wedge V_3 \rightarrow stay \end{cases}. \tag{4}$$

From $\mathcal{P}$, we can see that V_1 is a preferable policy for customers with characteristics U. However, if for some reason, policy V_1 is inapplicable to the specific customer group, $\mathcal{P}$ is no longer actionable in that it provides little knowledge on how to prevent the customers from switching to another company. Fortunately, rule cluster $\mathcal{C}$ suggests that another policy V_3 can be employed to retain those customers.

4.2 Interestingness Measures for Combined Patterns

Interestingness of Combined Association Rules. Traditional interestingness measures contribute little to selecting actionable combined patterns, because they are limited to the traditional simple association rules. Based on traditional supports, confidences and lifts, two new lifts are designed as follows for measuring the interestingness of combined association rules.

$$Lift_U(U \wedge V \rightarrow T) = \frac{Conf(U \wedge V \rightarrow T)}{Conf(V \rightarrow T)} = \frac{Lift(U \wedge V \rightarrow T)}{Lift(V \rightarrow T)} \tag{5}$$

$$Lift_V(U \wedge V \rightarrow T) = \frac{Conf(U \wedge V \rightarrow T)}{Conf(U \rightarrow T)} = \frac{Lift(U \wedge V \rightarrow T)}{Lift(U \rightarrow T)} \tag{6}$$

$Lift_U(U \wedge V \rightarrow T)$ is the lift of U with V as a precondition, which shows how much U contributes to the rule. Similarly, $Lift_V(U \wedge V \rightarrow T)$ gives the contribution of V in the rule. Based on the above two new lifts, the interestingness of combined association rules is defined as

$$I_{\text{rule}}(U \wedge V \rightarrow T) = \frac{Lift_U(U \wedge V \rightarrow T)}{Lift(U \rightarrow T)}. \tag{7}$$

It's easy to get

$$I_{\text{rule}}(U \wedge V \rightarrow T) = \frac{Lift(U \wedge V \rightarrow T)}{Lift(U \rightarrow T)\, Lift(V \rightarrow T)} \tag{8}$$

$$= \frac{Lift_V(U \wedge V \rightarrow T)}{Lift(V \rightarrow T)}. \tag{9}$$

I_{rule} indicates whether the contribution of U (or V) to the occurrence of T increases with V (or U) as a precondition. Therefore, "$I_{\text{rule}} < 1$" suggests that $U \wedge V \rightarrow T$ is less interesting than $U \rightarrow T$ and $V \rightarrow T$. The value of I_{rule} falls in $[0,+\infty)$. When $I_{\text{rule}} > 1$, the higher I_{rule} is, the more interesting the rule is.

I_{rule} works similarly as direction setting (DS) rules proposed by Liu et al. [6]. The difference is that their method gives an qualitative judgement on a rule whether it is a DS rule or not, while I_{rule} is a quantitative measure of the interestingness of a rule. I_{rule} measures how much is the unexpectedness of a combined rule against traditional simple association rules.

Interestingness of Combined Rule Pairs and Clusters. Suppose that $\mathcal{P}$ is a combined rule pair composed of R_1 and R_2 (See Formula 2), the interestingness of the rule pair $\mathcal{P}$ is defined as

$$I_{\text{pair}}(\mathcal{P}) = Lift_V(R_1)\, Lift_V(R_2)\, dist(T_1, T_2), \tag{10}$$

where $dist(\cdot)$ denotes the dissimilarity between two descendants. It is sometimes written as $I_{\text{pair}}(R_1, R_2)$ in this paper. For class with nominal values, such as "Pass" and "Fail", the dissimilarity can be defined as zero for two same descendants and as 1 for two different descendants. For ordinal class levels, such as

"Outstanding, Excellent, Good, Satisfactory, Fail", the similarity between "Outstanding" and "Fail" can be set to 1 and that between "Excellent" and "Good" can be set to 0.25. I_{pair} measures the contribution of the two different parts in antecedents to the occurrence of different classes in a group of customers with the same demographics or the same transaction patterns. Such knowledge can help to design business campaigns and improve business process. The value of I_{pair} falls in $[0,+\infty)$. The larger I_{pair} is, the more interesting a rule pair is.

For a rule cluster $\mathcal{C}$ composed of n combined association rules $R_1, R_2, \ldots, R_n$, its interestingness is defined as

$$I_{\text{cluster}}(\mathcal{C}) = \max_{i \neq j, R_i, R_j \in \mathcal{C}, T_i \neq T_j} I_{\text{pair}}(R_i, R_j). \tag{11}$$

The definition of I_{cluster} that we have provided indicates that interesting clusters are the rule clusters with interesting rule pairs, and the other rules in the cluster provide additional information. Same as I_{pair}, the value of I_{cluster} also falls in $[0,+\infty)$.

The interestingness of combined rule pair and cluster is decided by both the interestingness of rules and the most contrasting rules within the pair/cluster. A cluster made of contrasting confident rules is interesting, because it explains why different results occur and what to do to produce an expected result or avoid an undesirable consequence.

Selecting Combined Patterns. With the above interestingness measures, actionable combined patterns will be selected. First, the interesting combined rules are selected from the learned rules with support, confidence, lift, $Lift_U$, $Lift_V$ and I_{rule}. Second, the rules with high support and confidence are organized into pairs and then the pairs are ranked with I_{pair} to find contrasting rule pairs. Finally, related rules are added to selected rule pairs to build rule clusters.

Combined patterns are "actionable" in that: 1) for a single rule, $Lift_v$ measures the contribution of V to the result, which may suggest that V can be used to produce an expected outcome; and 2) the difference in the left hand of contrasting rules within a cluster explains why different results occur and how to get an expected result or avoid an undesirable consequence.

4.3 Redundancy in Combined Patterns

There are two kinds of redundancy in combined patterns: 1) the redundancy of combined rules within a rule cluster, and 2) the redundancy of combined rule pairs, which are defined as follows.

Definition 4 (Redundant Combined Association Rule). *Let $\mathcal{C}$ be a combined association rule cluster, and $R : U \wedge V \rightarrow T$ and $R' : U \wedge V' \rightarrow T'$ be two combined rules in $\mathcal{C}$, $R \in \mathcal{C}$, $R' \in \mathcal{C}$. R is redundant if $V' \subseteq V$, $T' = T$, $Lift(R') \geq Lift(R)$, $Lift_U(R') \geq Lift_U(R)$, $Lift_V(R') \geq Lift_V(R)$ and $I_{\text{rule}}(R') \geq I_{\text{rule}}(R)$.*

Definition 5 (Redundant Combined Rule Pair). *A combined rule pair $\mathcal{P}$ is redundant if: 1) there exists a rule pair $\mathcal{P}'$ with $I_{\text{pair}}(\mathcal{P}') \geq I_{\text{pair}}(\mathcal{P})$; and 2)*

for each $R : U \wedge V \rightarrow T \in \mathcal{P}$*, there exists a rule* $R' : U' \wedge V' \rightarrow T' \in \mathcal{P}'$ *with* $U' \subseteq U$*,* $V' \subseteq V$*,* $T' = T$*,* $Lift(R') \geq Lift(R)$*,* $Lift_U(R') \geq Lift_U(R)$*,* $Lift_V(R') \geq Lift_V(R)$ *and* $I_{\text{rule}}(R') \geq I_{\text{rule}}(R)$*.*

Our method for removing the two kinds of redundancy of combined patterns is composed of the following two steps.

1. Removing redundant rules in each rule cluster. This step is similar to the traditional way of removing redundant association rules, but only the redundancy within each rule cluster is removed here. Within each rule cluster $\mathcal{C}$ with the same U, each rule $R \in \mathcal{C}$ is checked to see whether there exist a rule R' in the same cluster with the same T and greater confidence, Lift, $Lift_U$, $Lift_V$ and I_{rule} and $V' \subseteq V$. If yes, then R is removed from $\mathcal{C}$ as a redundant rule.
2. Pruning redundant rule pairs. This step reduces the number of rule pairs. For two rule pairs $\mathcal{P}$ and $\mathcal{P}'$, if, for each rule $R \in \mathcal{P}$, there exists a rule $R' \in \mathcal{P}'$ with the same T and greater confidence, Lift, $Lift_U$, $Lift_V$ and I_{rule}, where U' and V' in R' is are respectively subsets of U and V in R, then all the rules in $\mathcal{P}$ are redundant with respect to $\mathcal{P}'$, and $\mathcal{P}$ is a redundant rule pair in terms of $\mathcal{P}'$. So $\mathcal{P}$ will be removed to reduce the number of rule pairs.

5 A Case Study

The technique we propose was tested with real-life data in Centrelink, a Commonwealth Government agency delivering a range of services to the Australian community. The data used was customer debts raised in calendar year 2006 and corresponding customer circumstances data and transactional arrangement / repayment data in the same year. The cleaned sample data contained 355,800 customers and their demographic attributes, as well as individual debt repayment arrangements. The aim was to find the association between demographics, arrangement/repayment methods and the class of customers, which could be used to recover debts as early as possible.

We discovered combined patterns in four steps. Firstly, the transactional data (with arrangements and repayments) was mined for frequent patterns. Secondly, the whole population was partitioned into groups by frequent transactional patterns. Thirdly, the demographic data of each customer group was mined for association rules. And lastly, combined patterns were generated by combining the above results. The minimum support was set to 20 (in the count of customers instead of percentage) and the minimum confidence was set to 60%. To discover interesting combined rules, we set $Lift > 1$, $Lift_U > 1$, $Lift_V > 1$, $I_{\text{pair}} > 1$ and $I_{\text{rule}} > 1$, and to discover interesting combined rule clusters, the selected rules were organized into clusters, with the rule clusters then ranked by I_{cluster}.

Generally speaking, to prune redundancy in association rules, when two rules have the same confidence and one rule is more general than the other, preference was given to the shorter one. Nevertheless, when analyzing the rules discovered in this exercise, we found that because some rules were on almost the same

Table 6. Traditional Association Rules

V		T	$Conf(\%)$	$Count$	$Lift$
Arrangement	Repayment	Class			
irregular	cash or post office	A	82.4	4088	1.8
withholding	cash or post office	A	87.6	13354	1.9
withholding & irregular	cash or post office	A	72.4	894	1.6
withholding & irregular	cash or post office & withholding	B	60.4	1422	1.7

Table 7. Selected Combined Rules

Rules	U	V		T	Cnt	$Conf$ (%)	I_r	$Lift$	L_U	L_V	$Lift$ of $U \rightarrow T$	$Lift$ of $V \rightarrow T$
	Demographics	Arrangement	Repayment	Class								
r_1	age:65+	withholding & irregular	withholding	C	50	63.3	2.91	3.40	2.47	4.01	0.85	1.38
r_2	income:0 & remote:Y & marrital:sep & gender:F	withholding	cash or post & withholding	B	20	69.0	1.47	1.95	1.34	2.15	0.91	1.46
r_3	income:0 & age:65+	withholding	cash or post & withholding	A	1123	62.3	1.38	1.35	1.72	1.09	1.24	0.79
r_4	income:0 & gender:F & benefit:P	withholding	cash or post	A	469	93.8	1.36	2.04	1.07	2.59	0.79	1.90

group of customers, business experts tended to prefer longer rules which provided more detailed information concerning the overall characteristics of the group. Therefore, in this case study, those rules with confidence less than 1.05 times that of more specific rules were removed as redundant rules, and the same was done to remove redundant rule clusters.

There were 7,711 association rules before removing redundancy of combined rules. After removing redundancy of combined rules, 2,601 rules were left, which built up 734 combined rule clusters. After removing redundancy of combined rule clusters, 98 rule clusters with 235 rules remained, which was within the capability of human beings to read. The traditional association rules we discovered from transactional data are given in Table 6. Some selected combined patterns are shown respectively in Tables 7 and 8. In the two tables, columns L_U and L_V stand for $Lift_U$ and $Lift_V$, respectively.

In Table 7, r_1: "*Age:65+, arrangement=withholding and irregular, repayment=withholding* $\rightarrow$ *C*" has a high I_{rule} of 2.91. "*Lift of* $U \rightarrow C$" indicate that the lifts of "*Age:65+* $\rightarrow$ *C*" is 0.85, which suggests that "*Age:65+*" is negatively associated with "C". However, $Lift_U = 2.47$ suggests that, under "*arrangement=withholding and irregular, repayment=withholding*", "*Age:65+*" becomes positively associated with "C". Moreover, $Lift_V$ is greater than "*Lift of* $V \rightarrow C$", which suggests that the contribution of the specific arrangement and repayment to the occurrence of "C" also increases in customer group "*Age:65+*". What's more, $Lift = 3.40$ also suggests that the combination of "*Age:65+*" and "*arrangement=withholding and irregular, repayment=withholding*" more than triples the probability of the occurrence of "C". Therefore, r_1 is a very interesting rule, which explains why it has a high value of I_{rule}. In contrast, r_5 in Table 8 has an I_{rule} of 0.86 (shown as I_r), which indicates that it is not interesting as a single

Table 8. Selected Combined Rule Clusters

Clusters	Rules	U	V		T	Cnt	$Conf$ (%)	I_r	I_c	$Lift$	L_U	L_V	$Lift$ of $U \to T$	$Lift$ of $V \to T$
		demographic	arrangement	repayment										
R_1	r_5	age:65+	withhold	cash or post	A	1980	93.3	0.86	6.5	2.02	1.06	1.63	1.24	1.90
	r_6		irregular	cash or post	A	462	88.7	0.87		1.92	1.08	1.55	1.24	1.79
	r_7		withhold & irregular	cash or post	A	132	85.7	0.96		1.86	1.18	1.50	1.24	1.57
	r_8		withhold & irregular	withhold	C	50	63.3	2.91		3.40	2.47	4.01	0.85	1.38
R_2	r_9	marital:sin	irregular	cash or post	A	400	83.0	1.12	6.3	1.80	1.01	2.00	0.90	1.79
	r_{10}	&gender:F	withhold	cash or post	A	520	78.4	1.00		1.70	0.89	1.89	0.90	1.90
	r_{11}	&benefit:N	withhold & irregular	cash or post & withhold	B	119	80.4	1.21		2.28	1.33	2.06	1.10	1.71
	r_{12}		withhold	cash or post & withhold	B	643	61.2	1.07		1.73	1.19	1.57	1.10	1.46
	r_{13}		withhold & vol. deduct	withhold & direct debit	B	237	60.6	0.97		1.72	1.07	1.55	1.10	1.60
	r_{14}		cash	agent	C	33	60.0	1.12		3.23	1.18	3.07	1.05	2.74
R_3	r_{15}	income:0	irregular	cash or post	A	191	76.7	1.03	5.1	1.66	0.93	1.85	0.90	1.79
	r_{16}	&age:22-25	cash	cash or post	C	440	62.1	1.08		3.34	1.31	2.76	1.21	2.56
R_4	r_{17}	benefit:Y	irregular	cash or post	A	218	79.6	1.15	4.1	1.73	0.97	2.06	0.84	1.79
	r_{18}	&age:22-25	cash	cash or post	C	483	65.6	0.78		3.53	1.38	1.99	1.78	2.56

rule. Although r_5 has a high lift of 2.02, its $Lift_U$ and $Lift_V$ are respectively less than "*Lift of* $U \to C$" and "*Lift of* $V \to C$", which suggests that the contribution of U and V to the occurrence of C becomes less when they are combined together. That is, for r_5, $U \wedge V \to C$ is actually less interesting or useful than $U \to C$ and $V \to C$. Nevertheless, it does not necessarily mean that r_5 is not interesting as a part of a rule cluster, since I_{rule} measures the interestingness of a single rule, not that of a rule cluster.

Some selected rule clusters are shown in Table 8. The clusters are ordered descendingly by I_{cluster} (shown as I_c). Within each cluster, the rules are ordered first ascendingly by class and then descendingly by $Lift_V$ (shown as L_V). For customers with "*maritle:single, gender:F, benefit:N*" (see R_2), "*Arrangement=irregular or withholding, Repayment=cash or post office*" is associated with *class A* (see r_9 and r_{10}), while "*Arrangement=cash, Repayment=agent recovery*" is associated with *class C* (see r_{14}). Here, *Class A* is preferable than *Class B*, and *Class B* is preferable than *Class C*. Therefore, for a single female customer with a new debt, if her benefit type is N, she may be encouraged to repay under "*Arrangement=irregular or withholding, Repayment=cash or post office*", and be persuaded not to repay under "*Arrangement=cash, Repayment=agent recovery*". In such a way, her debt will probably be repaid more quickly. For the above customer group of single female on benefit N, the priority of arrangement-repayment methods is given by the rules from r_9 to r_{14}. Such kind of knowledge is actionable in that it can help to improve policy or design campaigns to recover debts as soon as possible.

6 Conclusions

This paper presents a new idea of combined patterns. The concepts of combined association rules, combined rule pairs and combined rule clusters are defined; the

interestingness of each is designed; and two kinds of redundancy are analyzed. The proposed combined patterns are more useful and actionable than traditional simple association rules. And our technique, which has been tested with real-world data, has provided some interesting and helpful results.

Acknowledgments

Our thanks go to Mr. Fernando Figueiredo, Mr. Peter Newbigin and Mr. Brett Clark from Business Integrity Programs Branch, Centrelink, Australia, for their domain knowledge and helpful suggestions.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proc. of the ACM SIGMOD International Conference on Management of Data, Washington, D.C., USA, May 1993, pp. 207–216 (1993)
2. Hilderman, R.J., Carter, C.L., Hamilton, H.J., Cercone, N.: Mining Market Basket Data Using Share Measures and Characterized Itemsets. In: Wu, X., Kotagiri, R., Korb, K.B. (eds.) PAKDD 1998. LNCS, vol. 1394, pp. 159–170. Springer, Heidelberg (1998)
3. Lent, B., Swami, A.N., Widom, J.: Clustering association rules. In: Proceedings of the 13th International Conference on Data Engineering, Birmingham, U.K, April 7–11, pp. 220–231. IEEE Computer Society, Los Alamitos (1997)
4. Liu, B., Hsu, W.: Post-analysis of learned rules. In: Proceedings of the 13th National Conference on Artificial Intelligence (AAAI 1996), Portland, Oregon, USA, pp. 828–834 (August 1996)
5. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD 1998), pp. 80–86. AAAI Press, Menlo Park (1998)
6. Liu, B., Hsu, W., Ma, Y.: Pruning and summarizing the discovered associations. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 1999), pp. 125–134. ACM Press, New York (1999)
7. Zaïane, O.R., Antonie, M.-L.: On pruning and tuning rules for associative classifiers. In: Khosla, R., Howlett, R.J., Jain, L.C. (eds.) KES 2005. LNCS (LNAI), vol. 3683, pp. 966–973. Springer, Heidelberg (2005)

Efficient Single-Pass Mining of Weighted Interesting Patterns

Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer,
Byeong-Soo Jeong, and Young-Koo Lee

Department of Computer Engineering, Kyung Hee University
1 Seochun-dong, Kihung-gu, Youngin-si, Kyunggi-do, 446-701, Republic of Korea
{farhan,tanbeer,jeong,yklee}@khu.ac.kr

Abstract. Mining weighted interesting patterns (WIP) [5] is an important research issue in data mining and knowledge discovery with broad applications. WIP can detect correlated patterns with a strong weight and/or support affinity. However, it still requires two database scans which are not applicable for efficient processing of the real-time data like data streams. In this paper, we propose a novel tree structure, called SPWIP-tree (Single-pass Weighted Interesting Pattern tree), that captures database information using a single-pass of database and provides efficient mining performance using a pattern growth mining approach. Extensive experimental results show that our approach outperforms the existing WIP algorithm. Moreover, it is very efficient and scalable for weighted interesting pattern mining with a single database scan.

Keywords: Data mining, knowledge discovery, weighted interesting pattern mining, data stream, correlated patterns.

1 Introduction

Data mining discovers hidden and potentially useful information from databases. Frequent pattern mining [1], [2], [9], [11],[23] plays an important role in data mining and knowledge discovery techniques such as association rule mining, classification, clustering, time-series mining, graph mining, web mining etc. However, frequent pattern mining does not consider different weights for different items. Weighted frequent pattern mining approaches [3], [4], [5], [6],[7],[8],[22] have been proposed to overcome this problem.

Extensive growth of data gives the motivation to find meaningful weighted frequent patterns among the huge data. Yun [5] proposed weighted interesting patterns (WIP) by using a new measure, called weight-confidence, to mine correlated weighted frequent patterns with strong weight affinity. Weight-confidence prevents the generation of patterns with substantially different weight levels. In that research work [5], a weight range is used to decide weight boundaries and an h-confidence [20] is used to identify strong support affinity patterns. WIP not only gives a balance between the two measures of weight and support, but also considers weight affinity and/or support affinity between items within patterns

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 404–415, 2008.

so more correlated patterns can be detected [5]. For example, in a retail market, it prunes the patterns like {bronze ring, gold necklace}, {jeans pant, pen} etc. But it generates interesting patterns like {gold ring, gold necklace}, {trouser, T-shirt}, {pen, pencil, eraser} etc. Therefore, WIP algorithm is useful for generating weighted interesting patterns very quickly compared to the traditional weighted frequent pattern mining algorithms [3], [6], [7], [8] by pruning the non-correlated patterns at the early stage in the mining process. However, it still needs two database scans and therefore not suitable for efficiently processing of the real-time data like data streams.

A data stream, where data flows in the form of continuous stream, is a continuous, unbounded and ordered sequence of items that arrive in order of time. To find weighted interesting patterns from data streams, we no longer have the luxury of performing multiple data scans. Once the streams flow through, we lose them. In recent years, many applications generate data streams in real time, such as sensor data generated from sensor networks, transaction flows in retail chains, web record and click streams in web applications, performance measurement in network monitoring and traffic management, call records in telecommunications, and so on.

Motivated from these real world scenarios, we develop a single-pass approach to mine weighted interesting patterns. In this paper, we propose a novel tree structure SPWIP-tree (Single-pass Weighted Interesting Pattern tree), that captures database information using a single-pass of database without any restructuring operation. It exploits the pattern growth mining approach [9] to avoid the level-wise candidate generation-and-test [1], [2] problem. Our comprehensive experimental results on both real-life and IBM synthetic datasets show that our SPWIP-tree outperforms the WIP algorithm.

The remainder of this paper is organized as follows. In Section 2, we describe background and related work. In Section 3, we develop our proposed tree structure SPWIP-tree for weighted interesting pattern mining. Here, we also show the mining process of the SPWIP-tree. In Section 4, our experimental results are presented and analyzed. Finally, in Section 5, conclusions are drawn.

2 Background and Related Work

2.1 Preliminaries

We have adopted similar definitions presented in the previous works [3], [4], [5].

Definition 1 (*Weight of a pattern*). A weight of an item is a non-negative real number which is assigned to reflect the importance of each item in the transaction database. For a set of items $I = \{i_1, i_2, ...i_n\}$, weight of a pattern $P\{x_1, x_2, ...x_m\}$ is given as follows:

$$Weight(P) = \frac{\sum_{q=1}^{length(P)} Weight(x_q)}{length(P)} \quad (1)$$

Table 1. An example of transaction database with weight table

TID	Transaction
T_1	a,b,c,d,g
T_2	a,b,d,f
T_3	a,b,c,d,g,h
T_4	c,f,g
T_5	a,b,c,d
T_6	d,f,h

(a) Transaction database

Item	W
a	0.6
b	0.5
c	0.3
d	0.3
f	0.2
g	0.5
h	0.4

(b) Weight table

Definition 2 (*Weighed support of a pattern*). A weighted support of a pattern is defined as the resultant value of multiplying the pattern's support with the weight of the pattern. So the weighted support of a pattern P is given as follows:

$$Wsupport(P) = Weight(P) \times Support(P) \tag{2}$$

Definition 3 (*Weighed frequent pattern*). A pattern is called a weighted frequent pattern if the weighted support of the pattern is greater than or equal to the minimum threshold.

Example 1. Consider the example database in Table 1(a) and the weight for each item in Table 1(b). Pattern *"ab"* has support 4, as it occurred in T_1, T_2, T_3 and T_5. According to the equation 1 and 2, $Weight(ab) = (0.6 + 0.5)/2 = 0.55$ and $Wsupport(ab) = 0.55 \times 4 = 2.2$ respectively. If the *minimum Wsupport (min_Wsup)* is 1.6, pattern *"ab"* is a weighted frequent pattern according to definition 3.

Definition 4 (*Weight confidence of a pattern*). The weight confidence of a pattern $P\{x_1, x_2, ...x_m\}$, is denoted as *Wconf(P),* is a measure that reflects the overall weight affinity among items within the pattern. This measure is defined as

$$Wconf(P) = \frac{Min_{1 \le j \le m}(Weight\{x_j\})}{Max_{1 \le k \le m}(Weight\{x_k\})} \tag{3}$$

Definition 5 (*Hyperclique confidence of a pattern*). The hyperclique confidence of a pattern $P\{x_1, x_2, ...x_m\}$, is denoted as *Hconf(P),* is a measure to calculate support affinity among items within a pattern. This measure is defined as

$$Hconf(P) = \frac{Support\{x_1, x_2, ...x_m\}}{Max_{1 \le k \le m}(Support\{x_k\})} \tag{4}$$

Definition 6 (*Weighted interesting pattern*). A pattern is defined as a weighted interesting pattern if

1. The *Wsupport* of the pattern is no less than a *min_Wsup*
2. The *Wconf* of the pattern is no less than a *min_Wconf*
3. The *Hconf* of the pattern is no less than a *min_Hconf*

Example 2. Consider the example database in Table 1(a), the weight for each item in Table 1(b), *min_Wsup* = 1.6, *min_Wconf* = 0.6 and *min_Hconf* = 0.7. According to definition 4 and 5, pattern *"ab"* has *Wconf* = 0.5/0.6 = 0.833, and *Hconf* = 4/4 = 1 respectively. We have already calculated *Wsupport(ab)* = 2.2 in example 1. As a result, according to definition 6, it is a weighted interesting pattern. On the other hand, pattern *"ad"* has *Wsupport* = $\{(0.6+0.3)/2\} \times 4 = 0.45 \times 4 = 1.8$, *Wconf* = 0.3/0.6 = 0.5 and *Hconf* = 4/5 = 0.8. Therefore, it is a weighted frequent pattern but not a weighted interesting pattern (as its *Wconf* ≤ *min_Wconf*).

2.2 The Main Challenging Problem

The main challenging problem of weighted frequent pattern mining and weighted interesting pattern mining is, weighted frequency of an itemset (or a pattern) does not have the *downward closure* property [1], [2]. This property tells that if a pattern is infrequent then all of its super patterns must be infrequent Consider the example database in Table 1(a), the weight for each item in Table 1(b), *min_Wsup* = 1.8, *min_Wconf* = 0.6 and *min_Hconf* = 0.7. Pattern "*d*" is not a weighted interesting pattern, as *Wsupport(d)* = 0.3 × 5 = 1.5 ≤ 1.8. But its super pattern "*bd*" is a weighted interesting pattern as it has *Wsupport* = (0.5+0.3)/2×4 = 0.4×4 = 1.6, *Wconf* = 0.3 / 0.5 = 0.6 and *Hconf* = 4/5 = 0.8.

The *downward closure* property can be maintained by multiplying each pattern's frequency by the global maximum weight. In the above example, item "*a*" has the maximum weight of 0.6, and by multiplying it with the frequency of item "*d*", 3.0 can be obtained. So, pattern "*d*" is not pruned at the early stage and pattern "*bd*" is not missed. At the final stage, this overestimated pattern "*d*" is pruned finally by using its actual weighted frequency.

2.3 Related Work

The Apriori [1], [2] algorithm is the initial solution for the frequent pattern mining problem, but it suffers from the level-wise candidate generation-and-test problem and requires several database scans. FP-growth [9] solved this problem by using a FP-tree based solution without any candidate generation and using only two database scans. Some other research [10], [11], [16], [20], [21], [23] has been done for frequent pattern mining. The traditional frequent pattern mining considers an equal profit/weight for all items.

In the very beginning, some weighted frequent pattern mining algorithms MINWAL [6], WARM [7] and WAR [8] have been developed based on the Apriori [1], [2] algorithm using level-wise candidate generation-and-test mechanism. Obviously, these algorithms used multiple database scans and very slow in running time. WFIM [3] is the first FP-tree based weighted frequent pattern algorithm using two database scans and a static database. They have used a minimum weight and a weight range. Items are given fixed weights randomly from the weight range. They have arranged the FP-tree in the weight ascending order and they maintained the *downward closure* property on that tree. WLPMINER [4] algorithm finds weighted frequent patterns using length decreasing support constraints. WCloset [22] is proposed to calculate the closed weighted frequent patterns.

To find out not all the weighted frequent patterns but few weighted interesting patterns, a new measure weight-confidence is proposed to measure strong weight affinity patterns. This algorithm is known as weighted interesting pattern mining WIP [5] algorithm. They have used another measure hyperclique-confidence [20] to measure strong support affinity patterns. So, they have used actually three pruning conditions to find out these types of interesting patterns, (1) the weighted minimum threshold (which is used by WFIM), (2) the weight-confidence and (3) the hyperclique-confidence. However, WIP uses two database scans which are not suitable for stream data processing.

Some other mining algorithms [12], [13], [14], [15], [23] have been developed to find out frequent patterns over a data stream in real time. They used single-pass mining operation and showed that their approaches are quite efficient for frequent pattern mining using the recently available gigabyte range of memory. However, these solutions are not applicable for weighted interesting pattern mining. In this paper, we propose a single-pass approach for mining weighted interesting patterns.

3 Our Proposed Tree Structure

3.1 Construction

In this section, we describe the construction process of our proposed tree structure SPWIP-tree (Single-pass Weighted Interesting Pattern tree) using a single-pass of database. Header table is maintained in our tree structure to keep an item order. Each entry in a header table explicitly maintains item-id, frequency and weight information for each item. However, each node in a tree only maintains item-id and frequency information. To facilitate the tree traversals adjacent links are also maintained (not shown in the figures for simplicity) in our tree structure.

Consider the example database of Table 1. At first we create the header table and keep all the items in weight ascending order. After that we scan the transactions one by one, sort the items in a transaction according to the header table sort order and then insert into the tree. The first transaction T_1 has the items "a", "b", "c", "d", and "g". After sorting, the new order will be "c", "d", "g",

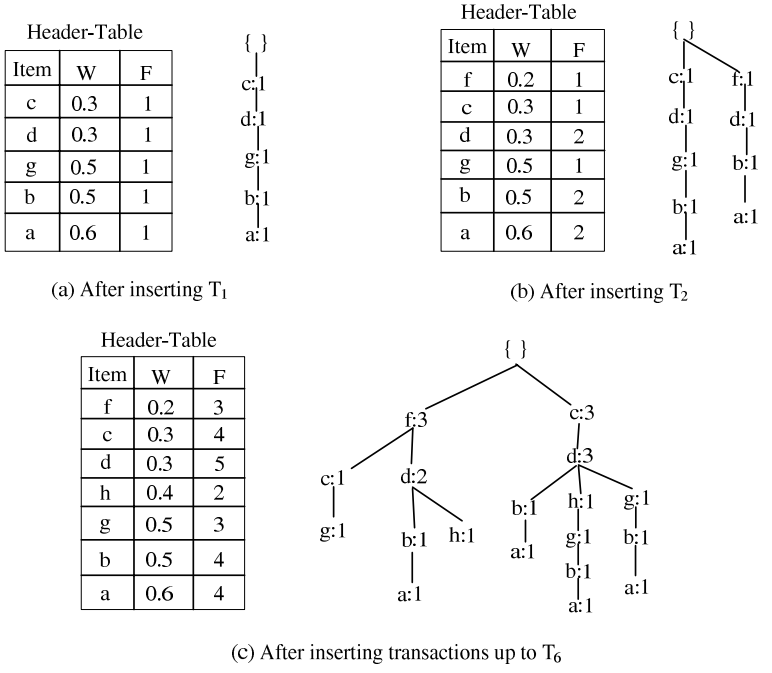


Fig. 1. SPWIP-tree construction

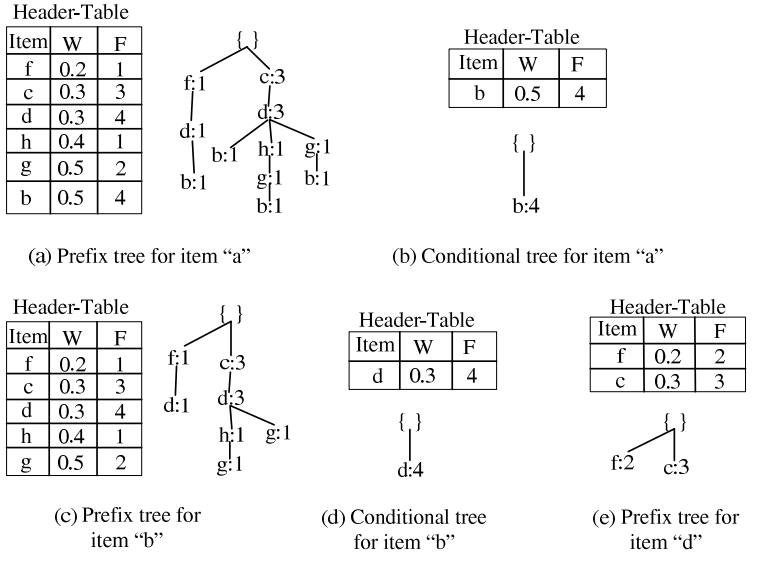
"b" and "a". Fig. 1(a) shows the SPWIP-tree after inserting T_1. Fig. 1(b) shows the SPWIP-tree after inserting T_2. In the same way all the transactions up to T_6 are inserted into the tree. Fig. 1(c) shows the final SPWIP-tree after inserting T_6. The following properties are true for the SPWIP-tree.

Property 1. The total count of frequency value of any node in the SPWIP-tree is greater than or equal to the sum of total counts of frequency values of its children.

Property 2. The SPWIP-tree can be constructed in a single-pass of database.

3.2 Mining Process

In this section, at first we describe the mining process of our proposed SPWIP-tree using a pattern growth approach. As discussed in Section 2.2, the main challenging problem in this area is, the weighted frequency of an itemset does not have the *downward closure* property and to utilize this property we have to use the global maximum weight. The global maximum weight, denoted by *GMAXW*, is the maximum weight of all the items in the whole database. For example, in Table 1(b), the item "a" has the global maximum weight of 0.6.



(a) Prefix tree for item "a"

(b) Conditional tree for item "a"

(c) Prefix tree for item "b"

(d) Conditional tree for item "b"

(e) Prefix tree for item "d"

Fig. 2. Mining process

Too many candidate patterns are generated if we use *GMAXW* throughout the mining process. By using the local maximum weight, denoted by *LMAXW*, we can reduce huge number of candidate patterns and also it is capable of maintaining the *downward closure* property. As SPWIP-tree is sorted in weight ascending order, *LMAXW* is reducing from bottom to top and the probability of a pattern to be a candidate is also reduced. This is one big advantage we can achieve in the bottom-up mining operation by using *LMAXW*. For example, after mining the weighted interesting patterns prefixing the item "*a*", when we go for the mining operation prefixing the item "*b*", then the item "*a*" will never come in any prefix/conditional trees. As a result, now we can easily assume that the item "*b*" has the maximum weight. This type of maximum weight in mining process is known as *LMAXW*. In Table 1, *LMAXW* for the bottom-most item "*a*" is 0.6 (which is also *GMAXW*). After that, *LMAXW* of the item "*b*" is 0.5 and in the same way we can calculate the *LMAXW* for the other items.

Consider the example database in Table 1(a), the weight for each item in Table 1(b), the SPWIP-tree constructed for that database in Fig. 1, $min\_Wsup = 2.0$, $min\_Wconf = 0.6$ and $min\_Hconf = 0.8$. Here the $GMAXW = 0.6$, and after multiplying the frequency of each item with *GMAXW*, the weighted frequency list is <*f*:1.8, *c*:2.4, *d*:3.0, *h*:1.2, *g*:1.8, *b*:2.4, *a*:2.4>. As a result, the candidate items are "*c*", "*d*", "*b*" and "*a*". Now we construct the prefix and conditional trees for these items in a bottom-up fashion and mine the weighted interesting patterns.

At first the prefix tree of the bottom-most item "*a*" (shown in Fig. 2(a)) is created by taking all the branches prefixing the item "*a*". We can eliminate the

Table 2. Candidate pruning in the first round

No.	Candidate patterns	*Wsupport* (maximum)	*Wconf*	*Hconf*	Result
1	*ac*	**0.6 × 3 = 1.8**	**0.3/0.6 = 0.5**	**3/4 = 0.75**	Pruned
2	*ad*	0.6 × 4 = 2.4	**0.3/0.6 = 0.5**	4/5 = 0.8	Pruned
3	*ab*	0.6 × 4 = 2.4	0.5/0.6 = 0.83	4/4 = 1	Pass
4	*bc*	**0.5 × 3 = 1.5**	0.3/0.5 = 0.6	**3/4 = 0.75**	Pruned
5	*bd*	0.5 × 4 = 2.0	0.3/0.5 = 0.6	4/5 = 0.8	Pass
6	*cd*	**0.3 × 3 = 0.9**	0.3/0.3 = 1	4/5 = 0.8	Pruned

Table 3. Candidate pruning in the final round

No.	Candidate patterns	*Wsupport* (actual)	Result
1	*ab*	({0.6+0.5}/2 × 4) = 2.2	Pass
2	*a*	0.6 × 4 = 2.4	Pass
3	*bd*	({0.5+0.3}/2 × 4) = 1.6	Pruned
4	*b*	0.5 × 4 = 2.0	Pass
5	*c*	0.3 × 4 = 1.2	Pruned
6	*d*	0.3 × 5 = 1.5	Pruned

global non-candidate items "*f*", "*h*" and "*g*" without any calculation. For item "*a*", $LMAXW = 0.6$ and we can get the weighted frequency list for the item "*a*" by multiplying the other item's frequency with *LMAXW*. Obviously this weighted frequency is the maximum possible weighted frequency of an itemset prefixing the item "*a*". So, we have to take all the patterns as a candidate having maximum weighted frequency greater than or equal to *min_Wsup*. Table 2 shows the candidate pruning in the first round. For the candidate patterns prefixing item "*a*", only the pattern "*ab*" can pass all the three tests. Therefore, conditional tree of the item "*a*" is created from its prefix tree by deleting all the nodes except the nodes containing the item "*b*". Four nodes of its prefix tree containing the item "*b*" are merged finally to form its conditional tree (shown in Fig. 2(b)). Candidate pattern "*ab*" is promoted here for the final step.

For the item "*b*" the $LMAXW = 0.5$ as the item "*a*" will not come out here. The prefix and conditional trees of the item "*b*" have been generated in the same way and shown in Fig. 2(c) and Fig. 2(d) respectively. Table 2 shows the candidate pattern "*bd*" is promoted for the final step. For item "*bd*" the $LMAXW = 0.3$ and its prefix tree is shown in Fig. 2(e). Table 2 shows that it does not have any candidate pattern to form its conditional tree. We have to test all the candidate patterns with their actual weights and the weighted frequency in the final round and mine the actual weighted interesting patterns. Table 3 shows the pruning process of the final round including the single-element patterns. Finally, the actual weighted interesting patterns are "*a*","*b*" and "*ab*".

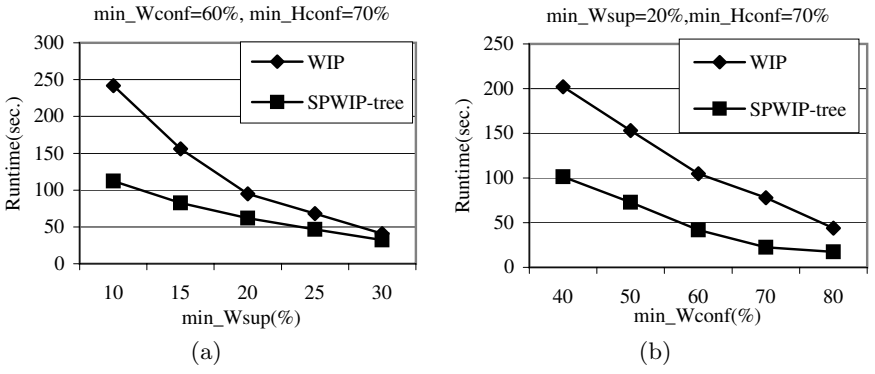


Fig. 3. Performance on the *mushroom* dataset

4 Experimental Results

To evaluate the performance of our proposed tree structure, we have performed several experiments on IBM synthetic dataset *T10I4D100K*, real-life *mushroom* and *kosarak* datasets [18], [19]. These datasets do not provide the weight values of each item. As like the performance evaluation of the previous weight based frequent pattern mining [3], [4], [5], [7], [8], [22] we have generated random numbers for the weight values of the items, ranging from 0.1 to 0.9. Our programs were written in Microsoft Visual C++ 6.0 and run with the Windows XP operating system on a Pentium dual core 2.13 GHz CPU with 1GB main memory.

The dataset *mushroom* contains 8,124 transactions and 119 distinct items. Its mean transaction size is 23, and it is a dense dataset. Fig. 3 shows the evaluation results for the *mushroom* dataset. We have given the performance differences by applying variable minimum weighted thresholds and minimum weight confidence thresholds. In Fig. 3(a), we have presented the performance result of our SPWIP-tree in comparison with the WIP algorithm using different minimum weighted thresholds. Minimum weight confidence and hyperclique confidence thresholds are 60% and 70% respectively. Similarly, Fig. 3(b) shows the performance comparison using different minimum weight confidences by using fixed minimum weighted threshold (20%)and hyperclique confidence threshold (70%). In all the cases, our SPWIP-tree outperforms the WIP algorithm by means of using a single database scan. However, the runtime difference is increasing when the minimum threshold is decreasing.

The *T10I4D100K* dataset contains 100,000 transactions and 870 distinct items. Its mean transaction size is 10.1, and it is a sparse dataset. Fig. 4(a) and 4(b) show the experimental results using variable minimum weighted threshold and minimum weight confidence threshold respectively. Like *mushroom* dataset, SPWIP-tree outperforms the WIP algorithm in all the cases.

We have tested the effectiveness of the SPWIP-Tree in stream data mining on the *kosarak* dataset. The dataset *kosarak* was provided by Ferenc Bodon

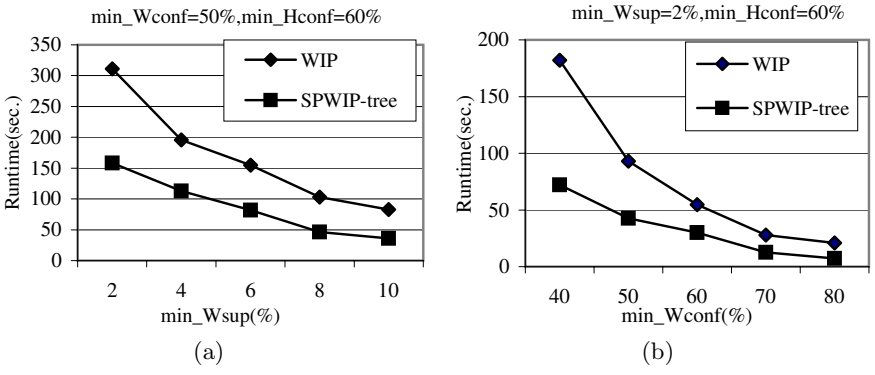


Fig. 4. Performance on the *T10I4D100K* dataset

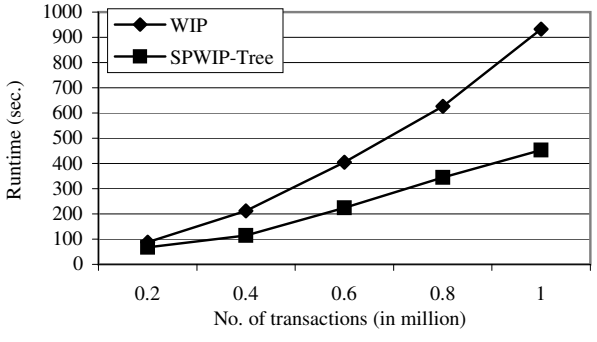


Fig. 5. Performance on the *kosarak* dataset

and contains click-stream data of a Hungarian on-line news portal [18], [19]. It contains 990,002 transactions and 41,270 distinct items. Its mean transaction size is 8.1, and it is a large sparse dataset. At first, we created the SPWIP-Tree for 0.2 million transactions of this dataset, and then performed a mining operation with $min\_Wsup = 5\%$, $min\_Wconf = 50\%$, and $min\_Hconf = 60\%$. Another 0.2 million transactions were added in the tree and the mining operations were performed again with the same minimum thresholds. In the same manner, all of the transactions in the *kosarak* dataset were added and the mining operation was performed in the SPWIP-Tree for each stage with the same minimum thresholds as shown in Fig. 5.

The existing WIP algorithm can not take any advantage from its tree structure when the dataset is increasing for stream data. For example, when the dataset size is 0.2 million transactions, tree structure of the WIP algorithm contains only the candidate patterns for the user given minimum thresholds. After that, when the dataset size is increased to 0.4 million transactions, it has to build its tree structure again from the very beginning using two database scans. Therefore,

the WIP algorithm is not efficient for stream data mining. Fig. 5 shows that SPWIP-Tree outperforms the WIP algorithm in stream data mining.

It is also shown in Fig. 5 that SPWIP-Tree has efficiently handled the 41,270 distinct items and around 1 million transactions in the real-life *kosarak* dataset. Therefore, this experimental result demonstrates the scalability of SPWIP-Tree to handle a large number of distinct items and transactions.

Many prefix tree based single-pass frequent mining research works [15], [16], [17], [21], [23] showed that the memory requirement for the prefix trees is low enough to use the gigabyte range memory now available. We have also handled our SPWIP-tree very efficiently within this memory range. Memory requirement of the SPWIP-tree constructed for the full *mushroom*, *T10I4D100K* and *kosarak* datasets are 0.692 MB, 14.285 MB and 148.762 MB respectively. Therefore, SPWIP-tree structure is efficient for the weighted interesting pattern mining using the recently available gigabyte range memory.

5 Conclusions

Weighted interesting pattern mining discovers very useful patterns with a strong weight and/or support affinity. However, the existing method WIP requires two database scans and therefore not applicable for efficient stream data mining. The main contribution of this paper is to provide a novel tree structure SPWIP-tree for single-pass weighted interesting pattern mining. It can handle the whole database information using a single scan of database and therefore applicable for stream data mining. It exploits a pattern growth mining approach to avoid the level-wise candidate generation-and-test problem. It is easy to construct and handle as it does not need any restructuring operation. Our comprehensive experimental results on both real-life and IBM synthetic datasets show that SPWIP-tree outperforms the existing WIP algorithm. Moreover, it is very efficient and scalable for single-pass weighted interesting pattern mining.

Acknowledgements. This study was supported by a grant of the Korea Health 21 R&D Project, Ministry For Health, Welfare and Family Affairs, Republic of Korea(A020602)

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM SIGMOD, pp. 207–216 (1993)
2. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: 20th Int. Conf. on Very Large Data Bases (VLDB), pp. 487–499 (1994)
3. Yun, U., Leggett, J.J.: WFIM: weighted frequent itemset mining with a weight range and a minimum weight. In: Fourth SIAM Int. Conf. on Data Mining, USA, pp. 636–640 (2005)
4. Yun, U., Leggett, J.J.: WLPMiner: Weighted frequent pattern mining with length decreasing support constraints. In: 9th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD), Vietnam, pp. 555–567 (2005)

5. Yun, U.: Efficient Mining of weighted interesting patterns with a strong weight and/or support affinity. Information Sciences 177, 3477–3499 (2007)
6. Cai, C.H., Fu, A.W., Cheng, C.H., Kwong, W.W.: Mining association rules with weighted items. In: Int. Database Engineering and Applications Symposium, IDEAS, pp. 68–77 (1998)
7. Tao, F.: Weighted association rule mining using weighted support and significant framework. In: 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, USA, pp. 661–666 (2003)
8. Wang, W., Yang, J., Yu, P.S.: WAR: weighted association rules for item intensities. Knowledge Information and Systems 6, 203–229 (2004)
9. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Mining and Knowledge Discovery 8, 53–87 (2004)
10. Grahne, G., Zhu, J.: Fast Algorithms for frequent itemset mining using FP-Trees. IEEE Transactions on Knowledge and Data Engineering 17(10), 1347–1362 (2005)
11. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery 15, 55–86 (2007)
12. Raissi, C., Poncelet, P., Teisseire, M.: Towards a new approach for mining frequent itemsets on data stream. Journal of Intelligent Information Systems 28, 23–36 (2007)
13. Jiang, N., Gruenwald, L.: Research Issues in Data Stream Association Rule Mining. SIGMOD Record 35(1), 14–19 (2006)
14. Metwally, A., Agrawal, D., Abbadi, A.E.: An Integrated Efficient Solution for Computing Frequent and Top-k Elements in Data Streams. ACM Transactions on Database Systems (TODS) 31(3), 1095–1133 (2006)
15. Leung, C.K.-S., Khan, Q.I.: DSTree: A Tree structure for the mining of frequent Sets from Data Streams. In: 6th IEEE Int. Conf. on Data Mining (ICDM), pp. 928–932 (2006)
16. Koh, J.-L., Shieh, S.-F.: An Efficient Approach for Maintaining Association Rules Based on Adjusting FP-tree Structures. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 417–424. Springer, Heidelberg (2004)
17. Li, X., Deng, Z.-H., Tang, S.: A fast algorithm for maintenance of association rules in incremental databases. In: Li, X., Zaïane, O.R., Li, Z. (eds.) ADMA 2006. LNCS (LNAI), vol. 4093, pp. 56–63. Springer, Heidelberg (2006)
18. Frequent itemset mining dataset repository, `http://fimi.cs.helsinki.fi/data/`
19. UCI machine learning repository, `http://kdd.ics.uci.edu/`
20. Xiong, H., Tan, P.-N., Kumar, V.: Hyperclique Pattern Discovery. Data Mining and Knowledge Discovery 13, 219–242 (2006)
21. Leung, C.K.-S., Khan, Q.I., Li, Z., Hoque, T.: CanTree: a canonical-order tree for incremental frequent-pattern mining. Knowledge and Information Systems 11(3), 287–311 (2007)
22. Yun, U.: Mining lossless closed frequent patterns with weight constraints. Knowledge-Based Systems 210, 86–97 (2007)
23. Tanbeer, S.K., Ahmed, C.F., Jeong, B.-S., Lee, Y.-K.: CP-tree: A tree structure for single-pass frequent pattern mining. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 1022–1027. Springer, Heidelberg (2008)

Pattern Taxonomy Mining for Information Filtering

Xujuan Zhou[1], Yuefeng Li[1], Peter Bruza[1],
Yue Xu[1], and Raymond Y.K. Lau[2]

[1] Faculty of Information Technology, Queensland University of Technology, Brisbane, QLD 4001, Australia
{x.zhou,y2.li,p.bruza,yue.xu}@qut.edu.au
[2] Department of Information Systems, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong
raylau@cityu.edu.hk

Abstract. This paper examines a new approach to information filtering by using data mining method. This new model consists of two components, namely, topic filtering and pattern taxonomy mining. The aim of using topic filtering is to quickly filter out irrelevant information based on the user profiles. The aim of applying pattern taxonomy mining techniques is to rationalize the data relevance on the reduced data set. Our experiments on Reuters RCV1(Reuters Corpus Volume 1) data collection show that more effective and efficient information access has been achieved by combining the strength of information filtering and data mining method.

1 Introduction

A personalized information access system fulfils the personalization information delivery by filtering out irrelevant information and/or identifying additional information of likely interest for the user based on the tailored user profiles. Most popular profiles are term-base profiles, such as probabilistic models [1], rough set models [2], BM25 [3] filtering models. The advantage of term-based profiles is efficient computational performance as well as mature theories for term weighting, which have emerged over the last couple of decades from the IR and machine learning communities. However, term-based profiles suffer from the problems of polysemy and synonymy. In addition, the threshold for accepting or rejecting documents is usually determined empirically. As IF systems are sensitive to data sets, it is a challenge to set the optimal threshold.

Many data mining techniques have been developed in order to underpin the IF system. However, the existing data mining techniques return numerous discovered patterns (knowledge) from a training set. Among these patterns, there are many meaningless patterns and many redundant patterns [4]. Pattern taxonomy model (PTM) [5] is a new approach of information gathering. Many up-to-date data mining techniques (e.g., sequential association rules, closed-pattern based non-redundant association rules) have been intergraded into this method. The

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 416–422, 2008.

PTM-based method used the pattern taxonomy rather than single word to represent user profiles and documents. However, pattern taxonomy mining is very sensitive to the noisy data (i.e. irrelevant documents).

To address the limitations of pattern taxonomy based approaches, a novel IF model which integrates topic filtering and pattern taxonomy mining strategies to provide more precise document filtering has been developed. The idea of integrating topic filtering into pattern taxonomy mining for information filtering has evolved from these two well established, but largely disparate fields. This proposed method intends to exploit the advantages of term-based approaches (IR) and pattern-based approaches (data mining) within the one system.

The remainder of the paper is organized as follows. Section 2 briefly introduces some related research works. Pattern taxonomy mining framework is then illustrated in Section 3. The empirical results will be reported in Section 4. The concluding remarks are given in Section 5.

2 Related Work

An Information Filtering (IF) [6] system monitors an incoming document stream to find the documents that match information needs of users. As the quality of the profiles directly influences the quality of information filtering, the issue of how to built accurate, reliable profiles is a crucial concern [7].

Traditional IF uses single-vector or multi-vector models which produce one term-weight [8] or more than one term-weight vectors [7] to represent the relevant information of the topic of likely interest for a user. There is a term "independence" assumption in those models. In contrast, discovering the association between a set of terms and a category (e.g., a term or a set of terms) is an important task in data mining. The current association discovery approaches include maximal patterns [9], sequential patterns [10], and closed patterns [11]. The main problem in most pattern discovery algorithms is that the system generates too many patterns while only a few of them are useful. Moreover, how to use the discovered patterns efficiently for decision making is another challenge.

3 Pattern Taxonomy Mining Framework

3.1 Basic Definitions

Let D be a training set of documents, which consists of a set of positive documents, D^+; and a set of negative documents, D^-. Let $T = \{t_1, t_2, \ldots, t_m\}$ be a set of terms (or keywords) which are extracted from the set of positive documents, D^+. A set of terms is referred to as a *termset*. Given a positive document d_i and a term t, $tf(d_i, t)$ is defined as the number of occurrences of t in d_i.

A set of term frequency pairs

$$p_{d_i} = \{(t, f) | t \in T, f = tf(t, d_i) > 0\}$$

is referred to as an initial *r-pattern* (rough pattern) in this paper.

Let $termset(p) = \{t|(t, f) \in p\}$ be the *termset* of p. In this paper, r-pattern p_1 equals to r-pattern p_2 if and only if $termset(p_1) = termset(p_2)$. A r-pattern is uniquely determined by its *termset*. Two initial r-patterns can be composed if they have the same *termset*. In this paper, the composition operation, $\oplus$, that defined in [12] is used to compose r-patterns.

Based on the above definitions, a set of composed r-patterns in D^+, can be obtained: $RP = \{p_1, p_2, \ldots, p_r\}$, where $r \leq n$, and $n = |D^+|$ is the number of positive documents in D. The support of a r-pattern p_i is the fraction of the initial r-patterns that are composed to form p_i.

Formally the relationship between r-patterns and terms can be described as the following *association mapping* if term frequencies are considered:

$$\beta : RP \rightarrow 2^{T \times [0,1]} - \{\emptyset\}, \text{ such that} \tag{1}$$

$$\beta(p_i) = \{(t_1, w_1), (t_2, w_2), \ldots, (t_k, w_k)\},$$

where $p_i = \{(t_1, f_1), (t_2, f_2), \ldots, (t_k, f_k)\} \in RP$ and $w_i = \frac{f_i}{\sum_{j=1}^{k} f_j}$, and $\beta(p_i)$ is called the *normal form* of r-pattern p_i in this paper. The association mapping β can derive a probability function for the term weight distribution on T in order to show the importance of terms in the positive documents, which satisfies:

$$pr_\beta(t) = \sum_{p_i \in RP, (t,w) \in \beta(p_i)} support(p_i) \times w, \text{ for all } t \in T \tag{2}$$

Now, a positive document d_i can be described as an event that represents what users want with the probability value: $pr(d_i) = \sum_{t \in d \cap T} pr_\beta(t)$.

3.2 Topic Filtering Stage

Pattern taxonomy mining (PTM [5])is sensitive to the noisy data. This is because of that patterns have low frequency of occurrence, a small *min_sup* has to be used in order to find interesting patterns. The consequence is that some noise terms and their combinations (patterns) are also retained and that make some negative documents obtain large weights in pattern mining model.

In response to this, a topic filtering model was developed. Based on the rough set theory [13], the decision rules for the partitioning of the incoming document stream into the positive, boundary, and negative regions have been developed for an IF model in [2]. The theory has been further developed for setting thresholds to determine relevant information in [14]and [12].

Let $d_i \in D^+$ be a positive document and d be an incoming document. The basic assumption for the rough-set based approach is that d is possibly relevant if $d_i \supseteq d$. The set of all document d such that $d_i \subseteq d$ is called the *covering set* of d_i.

The union of all covering sets of all $d_i \in D^+$ is called the *positive region* (POS) of incoming documents. The set of all documents d such that $\exists d_i \in D^+ \Rightarrow d_i \cap d \neq \emptyset$ is called the *boundary region* (BND). Also, the set of all

documents d such that $\forall d_i \in D^+ \Rightarrow d_i \cap d = \emptyset$ is called the *negative region* (NEG).

Given a document d, the decision rules can be determined ideally as follows:

$$\frac{\exists d_i \in D^+ \Rightarrow d_i \subseteq d}{d \in POS}, \text{ and} \frac{\exists d_i \in D^+ \Rightarrow d_i \cap d \neq \emptyset}{d \in BND}, \text{ and} \frac{\forall d_i \in D^+ \Rightarrow d_i \cap d = \emptyset}{d \in NEG}.$$

The theory also recommended a method to theoretically determine thresholds for finding relevant documents if there is a probability function pr for terms in T. For example, in [14], this function was defined as

$$pr(t) = \frac{1}{n} \sum_{d_i \in D^+, t \in d_i} \frac{1}{|d_i \cap T|}, \text{ for terms } t \in T$$

In the topic filtering stage, the positive documents in the training set have been represented as r-patterns and discovered r-patterns are employed to filter out most irrelevant documents rather than to identify relevant documents. The basic assumption is that document d is irrelevant if it is not closed to the common feature of the positive documents in the training set. Therefore, the threshold can determined as follows:

$$threshold = \overline{m} + \gamma(\sigma + skew) \tag{3}$$

where,$\overline{m}$ is the mean of the probabilities of positive documents in D^+, $\overline{m} = \frac{1}{n}\sum_{d_i \in D^+} pr(d_i)$; γ is an experimental coefficient; σ is the standard deviation of the probabilities of positive documents

$$\sigma = \sqrt{\frac{1}{n} \sum_{d_i \in D^+} (pr(d_i) - \overline{m})};$$

and *skew* is the skewness of the probabilities

$$skew = \frac{\sqrt{n}\sum_{d_i \in D^+} (pr(d_i) - \overline{m})^3}{(\sum_{d_i \in D^+} (pr(d_i) - \overline{m})^2)^{\frac{3}{2}}}.$$

3.3 Pattern Mining Stage

The concept of frequent and closed pattern [5] is also suitable for sequential patterns. A sequential pattern $s =< t_1, \ldots, t_r >$ ($t_i \subseteq T$) is an ordered list of terms. A sequence $s_1 =< x_1, \ldots, x_i >$ is a sub-sequence of another sequence $s_2 =< y_1, \ldots, y_j >$, denoted by $s_1 \subseteq s_2$, iff $\exists j_1, \ldots, j_y$ such that $1 \leq j_1 < j_2 \ldots < j_y \leq j$ and $x_1 = y_{j_1}, x_2 = y_{j_2}, \ldots, x_i = y_{j_y}$. Patterns can be organized into a taxonomy by using the is_a (or "subset") relation. We use the method developed by Wu and his colleagues to build a pattern taxonomy. For detail information about this method please refer to [5].

For calculating a specificity value for each pattern a term's support is needed to evaluated first. The evaluation of term supports (weights) is different to the

normal term-based approaches. In the term based approaches, a component of a given term's weighting is based on its appearance in documents. In the following, terms are weighted according to their appearance in discovered patterns.

Formally, for all positive document $d_i \in D^+$, its closed patterns are deployed onto a common set of terms T in order to obtain the following r-patterns:

$$\overrightarrow{d_i} =< (t_{i_1}, n_{i_1}), (t_{i_2}, n_{i_2}), \ldots, (t_{i_m}, n_{i_m}) > \quad (4)$$

where t_{i_j} in pair (t_{i_j}, n_{i_j}) denotes a single term and n_{i_j} is its *support* in d_i which is the number of closed patterns that contain t_{i_j}.

These r-patterns are composed by using an association mapping (see Eq. 2), and then the supports of the terms in D^+ are calculated.

To improve the efficiency of the mining of the pattern taxonomy, an algorithm, $SPMining$, was proposed in [5] to find all closed sequential patterns, which used the well known *Apriori* property in order to reduce the searching space.

Algorithm PTM2 describes the training process of pattern taxonomy mining. For every positive document, the SPMining algorithm is first called in step (2) giving rise to a set of closed sequential patterns SP. Additionally, all discovered patterns in a positive document are composed into an r-pattern giving rise to a set of r-patterns RP in step (2). Thereafter in step (3), the support is calculated for all terms that appear in the r-patterns. Finally, in step (4), the normal forms $\beta(p)$ for all r-pattern $p \in RP$ is used.

Algorithm PTM2 (D^+, *min_sup*)
Input: D^+; minimum support, *min_sup*.
Output: a set of r-patterns RP, and supports of terms.
Method:
(1) $RP = \emptyset$;
(2) **for** (document $d \in D^+$) {
  let DP be the set of paragraphes in d;
  //sequential pattern mining in a set of paragraphes
  SP = SPMining(DP, *min_sup*);
  $\overrightarrow{d} = \emptyset$;
  for (pattern $p_i \in SP$) {
    $p = \{(t, 1) | t \in p_i\}$;
    $\overrightarrow{d} = \overrightarrow{d} \oplus p$ }
  $RP = RP \cup \{\overrightarrow{d}\}$ }
(3) $T = \{t | (t, w) \in p, p \in RP\}$;
  for (term $t \in T$) $support(t) = 0$;
(4) **for** (r-pattern $p \in RP$)
  for ($(t, w) \in \beta(p)$)
    $support(t) = support(t) + w$;

After the support of terms have been computed from the training set, a given pattern's specificity to the given topic can be defined as follows:

$$spe(p) = \sum_{t \in p} support(t).$$

It is also easy to verify $spe(p_1) \leq spe(p_2)$ if p_1 is a sub-pattern of pattern p_2. This property shows that a document should be assigned a large weight if it contains large patterns. Based on this observation, the following weight will be assigned to a document d for ranking documents in pattern taxonomy model:

$$weight(d) = \sum_{t \in T} support(t)\tau(t, d).$$

4 Experimental Evaluation

Reuters Corpus Volume 1 (RCV1) was used to test the effectiveness of the proposed model. TREC (2002) has developed and provided 100 topics for the filtering track aiming at building a robust filtering system. All 100 topics have been used to represent the users's interested topics. The documents are treated as plain text documents by preprocessing the documents. The tasks of removing stop-words according to a given stop-words list and stemming term by applying the Porter Stemming algorithm are conducted. For each topic, 150 terms in the positive documents were chosen based on *if*idf* values for all term based models.

Table 1. Results of SVM, PTM and TSTM on 100 topics, where % means the percentage change in performance

	SVM	PTM	TSTM	%chg
b/p	0.450	0.454	**0.552**	+21.6
MAP	0.467	0.475	**0.574**	+20.8
$F_{\beta=1}$	0.452	0.453	**0.514**	+13.5

The new model is called Two-Stage Text Mining model (TSTM). Information filtering can also be regarded as a special instance of text classification [8]. SVM is a statistical method that can be used to find a hyperplane that best separates two classes. SVM achieved the best performance on the Reuters-21578 data collection for document classification [15]. SVM and the original PTM model [5] are our baseline models.

The standard measure such as F-beta ($F_\beta = 1$) measure, Mean Average Precision (MAP), the break-even point (b/p) measure were used to evaluate the results. The results for the IR/IF standard measure methods are shown in the table 1. As shown in Table 1, the performance of TSTM model is better than SVM and PTM model. Therefore, we conclude that the topic filtering stage is very useful for removing the "noises" in the incoming documents.

5 Conclusions

This paper illustrates a new model which integrates topic filtering and pattern taxonomy mining together to alleviate information overload and mismatch

problems. The proposed method has been evaluated using the standard TREC routing framework with encouraging results.

Compared with SVM and PTM based methods, the results of experiments on RCV1 collection demonstrate that the performance of information filtering can be significantly improved by the proposed new model. The substantial improvement is mainly due to the threshold applied to the topic filtering first stage and the "semantic" nature of patterns in the second stage. This research provides a promising methodology for using patterns in information filtering.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (1999)
2. Li, Y., Zhang, C., Swan, J.R.: An information filtering model on the web and its application in jobagent. Knowl.-Based Syst. 13(5), 285–296 (2000)
3. Robertson, S.E., Soboroff, I.: The trec 2002 filtering track report. In: TREC (2002)
4. Xu, Y., Li, Y.: Generating concise association rules. In: CIKM, pp. 781–790 (2007)
5. Wu, S.T., Li, Y., Xu, Y.: Deploying approaches for pattern refinement in text mining. In: ICDM, pp. 1157–1161 (2006)
6. Belkin, N.J., Croft, W.B.: Information filtering and information retrieval: two sides of the same coin? Commun. ACM 35(12), 29–38 (1992)
7. Mostafa, J., Mukhopadhyay, S., Lam, W., Palakal, M.J.: A multilevel approach to intelligent information filtering: Model, system, and evaluation. ACM Trans. Inf. Syst. 15(4), 368–399 (1997)
8. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. 34(1), 1–47 (2002)
9. Feldman, R., Aumann, Y., Amir, A., Zilberstein, A., Klösgen, W.: Maximal association rules: A new tool for mining for keyword co-occurrences in document collections. In: KDD, pp. 167–170 (1997)
10. Tzvetkov, P., Yan, X., Han, J.: Tsp: Mining top- closed sequential patterns. Knowl. Inf. Syst. 7(4), 438–457 (2005)
11. Han, J., Fu, Y.: Mining multiple-level association rules in large databases. IEEE Trans. Knowl. Data Eng. 11(5), 798–804 (1999)
12. Li, Y., Zhong, N.: Mining ontology for automatically acquiring web user information needs. IEEE Transactions on Knowledge and Data Engineering 18(4), 554–568 (2006)
13. Yao, Y., Wong, S.K.M.: A decision theoretic framework for approximating concepts. International Journal of Man-Machine Studies 37(6), 793–809 (1992)
14. Li, Y., Zhong, N.: Web mining model and its applications for information gathering. Knowledge-Based Systems 17, 207–217 (2004)
15. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: SIGIR, pp. 42–49 (1999)

An AI-Based Causal Strategy for Securing Statistical Databases Using Micro-aggregation

B. John Oommen[1] and Ebaa Fayyoumi[2]

[1] *Chancellor's Professor* ; *Fellow : IEEE* and *Fellow : IAPR*. This author can be contacted at : School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. This author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway
oommen@scs.carleton.ca

[2] This author can be addressed at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6.
efayyoum@scs.carleton.ca

Abstract. Although Artificial Intelligent (AI) techniques have been used in various applications, their use in maintaining security in Statistical DataBases ($SDBs$) has not been reported. This paper presents results, that to the best of our knowledge is pioneering, by which concepts from causal networks are used to secure SDBs. We consider the Micro-Aggregation Problem (MAP) in secure SDBs which involves partitioning a set of individual records in a micro-data file into a number of mutually exclusive and exhaustive groups. This problem, which seeks for the best partition of the micro-data file, is known to be NP-hard, and has been tackled using many heuristic solutions. In this paper, we would like to demonstrate that in the process of developing Micro-Aggregation Techniques ($MATs$), it is expedient to incorporate AI-based causal information about the dependence between the random variables in the micro-data file. This can be achieved by pre-processing the micro-data *before* invoking any MAT, in order to extract the useful dependence information from the joint probability distribution of the variables in the micro-data file, and then accomplishing the micro-aggregation on the "maximally independent" variables. Our results, on artificial life data sets, show that including such information will enhance the process of determining how many variables are to be used, and which of them should be used in the micro-aggregation process.

1 Introduction

The traditional methods used to ensure and preserve security and privacy in Statistical DataBases ($SDBs$) involves methods akin to the field of probability and statistical analysis. As far as we know the use of AI techniques to resolve these problems are unreported. In this paper, we present some pioneering AI-based *causal* methods to preserve privacy in $SDBs$. From a birds-eye perspective, this strategy works as follows: The relationship between the variables of the $SDBs$ are considered as nodes in a causal network. From this perspective we hypothesize that the inter variable dependence is crucial in determining the way to

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 423–434, 2008.

micro-aggregate the variables in the MAT. To achieve this, we involve an AI-based causal analysis to obtain the best dependence tree relating them. Once this information is gleaned, it is used to micro-aggregate the data, so that when the user requests information s/he is given as much information as absolutely crucial, but, is simultaneously, prevented from inferring the identity of the records associated with the data. The results what we have obtained on testing the method on artificial databases clearly demonstrate the power of our strategy.

A lot of attention has recently been dedicated to the problem of maintaining the confidentiality of statistical databases through the application of statistical tools, so as to limit the identification of information on individuals and enterprises. Statistical Disclosure Control (SDC) seeks a balance between the confidentiality and the data utility criteria. Therefore, optimizing the Information Loss (IL) and the Disclosure Risk (DR) so as to reach an equilibrium point between them is not an easy task [1].

Micro-aggregation is one of the most recent techniques that has been used to mask micro-data files with the intention of protecting them against re-identification in secure statistical databases [3].

The Micro-Aggregation Problem (MAP), as formulated in [3,6], can be stated as follows: A micro-data set $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ is specified in terms of the n "micro-records", namely the $U_i's$, each representing a data vector whose components are d continuous variables. Each data vector can be viewed as $U_i = [u_{i1}, u_{i2}, \ldots, u_{id}]^T$, where u_{ij} specifies the value of the j^{th} variable in the i^{th} data vector. Micro-aggregation involves partitioning the n data vectors into, say m, mutually exclusive and exhaustive groups so as to obtain a k-partition $\mathbb{P}_k = \{G_i \mid 1 \leq i \leq m\}$, such that each group, G_i, of size, n_i, contains either k data vectors or between k and $2k-1$ data vectors.

The optimal k-partition, $\mathbb{P}_k$, is defined to be the one that maximizes the within-group similarity, which is defined as the *Sum of Squares Error*, $SSE = \sum_{i=1}^{m} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^T (X_{ij} - \bar{X}_i)$. This quantity is computed on the basis of the Euclidean distance of each data vector X_{ij} to the centroid $\bar{X}_i$ of the group to which it belongs. The *Information Loss* is measured as $IL = \frac{SSE}{SST}$, where SST is the squared error that would result if all records were included in a single group, and is given as $SST = \sum_{i=1}^{m} \sum_{j=1}^{n_i} (X_{ij} - \bar{X})^T (X_{ij} - \bar{X})$, where $\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$.

As mentioned in the literature, this problem in its multi-variate setting is known to be NP-hard [8], and has been tackled using different approaches such as hierarchical clustering [3], genetic algorithms [3,13], graph theory [6,7], fuzzy clustering [14], machine learning [5], and neural network [9]. All the heuristic Micro-aggregation Techniques ($MATs$), seek to minimize the value of the IL. However, minimizing the loss in the data utility is an important issue, which is difficult to enforce, primarily because this strategy was intended to enhance the security in an SDC technique.

Understanding the presence and structure of dependency between a set of random variables is a fundamental problem in the design and analysis of many types of systems including filtering, pattern recognition etc. As far as we know its application in SDC has been minimal. Utilizing this information is the goal of this paper.

In general, the result of the multi-variate $MATs$ depends on the number of variables used in the micro-aggregation process. In other words, deciding on the number of variables to be taken into account, and on the *identity* of the variables to be micro-aggregated, is far from trivial, although some initial results and conjectures are given in [4,11]. The unanswered question is that of inferring which variables should be used in this process. We believe that a solution to this puzzle lies in the inter-variable "dependence" information. Our aim is to seek a systematic process by which we can choose the desired variables automatically and micro-aggregate the file. We propose a scheme by which we can determine the "maximally independent" variables in the subsequent multi-variate micro-aggregation.

The structure of this paper is as follows: In Section 2 the enhanced micro-aggregation dependence is presented informally. Then, in Section 3, we present the results of experiments we have carried out for artificial data sets. The paper finishes in Section 4 with some conclusions.

2 Enhancing Micro-aggregation with Dependence

It is well-known that the result of the multi-variate $MATs$ depends on the number and the *identity* of the variables used in the micro-aggregation process. Since multi-variate micro-aggregation using two or three variables at a time offers the best trade-off between the IL and the DR [4], we hope to minimize the time needed to evaluate the distance between a single micro-data record and the mean of the group it belongs to. This computation involves evaluating

$$D(X,Y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}. \tag{1}$$

where X and Y are two multi-variate data vectors with their components being $\{x_i\}$ and $\{y_i\}$ respectively, and d represents the dimension of the space. Searching over all possible subsets of variables is combinatorially hard.

As argued in the unbridged version of the paper, the key idea in choosing a sub-set of the variables by avoiding the combinatorial solution, should be based on the dependence model of the micro-data file. This will, in turn, reduce the dimensionality[1] of the space to $d' < d$. The new distance that will thus be computed will be:

$$D'(X,Y) = \sqrt{\sum_{i=1}^{d'}(x_i - y_i)^2} \qquad \text{where } d' < d. \tag{2}$$

The reduction in the dimensionality is achieved formally by maximally using the information in the almost independent variables, and we plan to do this by finding the best dependence tree [15,16].

[1] The reader should observe that our goal is quite distinct from the reported methods of projecting the multi-dimensional space onto a single axis using a particular variable, the sum z-scores scheme, or a principle component analysis.

We formalize these concepts. The joint probability distribution of the random vector $\mathbf{V} = [V_1, V_2, \ldots, V_d]^T$ in terms of conditional probabilities is given as

$$P(\mathbf{V}) = P(V_1)P(V_2|V_1)P(V_3|V_1, V_2)\ldots P(V_d|V_1, V_2, \ldots, V_{d-1}). \tag{3}$$

where each V_i is a random variable.

In (3) each variable is conditioned on an increasing number of other variables. Therefore, estimating the k^{th} term of this equation requires maintaining the estimates of all the k^{th} order marginals which is infeasible. We simplify the dependency model by restricting ourselves to second-order marginals as [15]:

$$P_a(\mathbf{V}) = \prod_{i=1}^{d} Pr(V_i|V_{j(i)}). \tag{4}$$

where $P_a(\mathbf{V})$ is the approximated form of $P(\mathbf{V})$, and V_i is conditioned on $V_{j(i)}$ for $0 \leq j(i) < i$.

The dependence of the variables can be represented as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{W})$ where $\mathbf{V} = \{V_1, V_2, \ldots, V_d\}$ is a finite set of vertices, which represents the set of random variables in the micro-data file with d dimensions, $\mathbf{E}$ is a finite set of edges $\{\langle V_i, V_j\rangle\}$, where $\langle V_i, V_j\rangle$ represents an edge between the vertices V_i and V_j. Finally, $\mathbf{W} = \{w_{i,j}\}$ is a finite set of weights, where $w_{i,j}$ is the weight assigned to the edge $\langle V_i, V_j\rangle$ in the graph. The values of these weights can be calculated based on a number of measures, as will be explained presently.

In $\mathbf{G}$, the edge between any two nodes represents the fact that these variables are statistically dependent [2]. In such a case, the weight, $w_{i,j}$, can be assigned to the edge as being equal to $I^*(V_i, Vj)$, the Expected Mutual Information Measure ($EMIM$), metric between them, which is:

$$I^*(V_i, V_j) = \sum_{v_i, v_j} Pr(v_i, v_j) \log \frac{Pr(v_i, v_j)}{Pr(v_i)Pr(v_j)}. \tag{5}$$

Any edge, say $\langle V_i, V_j\rangle$ with the edge weight $I^*(V_i, V_j)$ represents the fact that V_i is stochastically dependent on V_j, or that V_j is stochastically dependent on V_i. Although, in the worst case, any variable pair could be dependent, the model expressed by Eq.(4) imposes a tree-like dependence. It is easy to see that this graph includes a large number of trees (actually, an $\bigcirc(d^{(d-2)})$ of such spanning trees). Each of these trees represents a unique approximated form for the density function $P(\mathbf{V})$. Chow and Liu proved that searching for the best "dependence tree" is exactly equivalent to searching for the Maximum Spanning Tree (MST) of the graph [2]. Further, since the probabilities that are required for computing the edge weights are not known *a priori*, Valiveti and Oommen showed that this could be achieved by estimating them in a maximum likelihood manner [15,16]. They showed that the maximum likelihood estimate (ML) for the best dependence tree, can be obtained by computing the MST of the graph, where the edge weights are computed using the $EMIM$ of the estimated probabilities.

Since these dependent variables are maximally-correlated to the variable that they depend on, we propose to use the vertices that have the maximum number of In/Out edges in the graph to micro-aggregate the micro-file. We believe

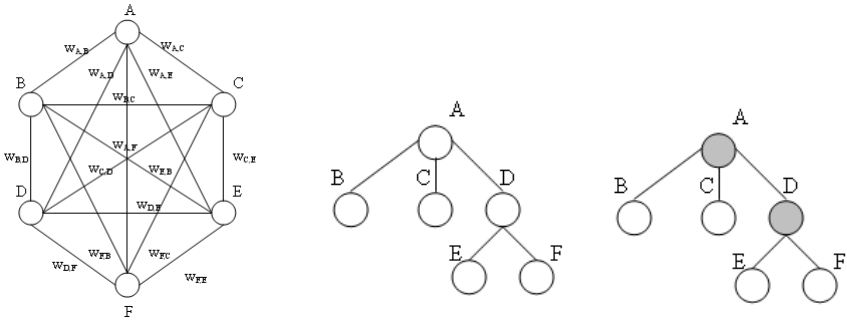


Fig. 1. (a) The fully-connected undirected graph represents the dependence between six random variables. (b) An example of a dependence tree used to micro-aggregate the data file containing 6 variables.

that the nodes which possess this property are the best candidates to reflect the characteristics of the entire multi-variate data set because they connect to the maximum number of nodes that statistically depend on it, as argued in Conjecture 1. The rationale for Conjecture is omitted here in the interest of brevity.

Conjecture 1. Micro-aggregating the micro-data file can be best achieved if the nodes which possess the maximum number of In/Out edges in the tree obtained as the MST of the underlying graph **G**, are used as an input to solve the MAT.

For example, consider the set of variables in Fig. 1, where we intend to micro-aggregate it using the Maximum Distance Average Vector (MDAV) method[2]. Fig. 1.a shows the fully connected graph, G, after computing the $EMIM$-based edge weights for all pairs of nodes. By using the strategy alluded to above, we obtain a tree as in Fig. 1.b, which shows the case when the MST leads to the ML condition, and Fig. 1.c shows that the selected sub-set of the variables.

3 Experimental Results

3.1 Data Sets

In order to verify the validity of our methodology in projecting the multi-variate data set into a subset of random variables to be used in the micro-aggregation process, three simulated data sets were used in the testing phase: (i) Sim_1 of 8 dimensions and 5000 records, (ii) Sim_2 of 16 dimensions and 10,000 records, (iii) Sim_3 of 22 dimensions and 22,000 records.

The simulated data were generated or tested for various dimensions of random vectors, as follows: First of all, the number of random variables was determined.

[2] The MDAV is the first algorithm to accomplish micro-aggregation without projecting the multi-variate data onto a single axis. It micro-aggregates the multi-variate micro-data file based on the concept of the centroid of the data set as explained in [3,12].

Thereafter, the "true" structure of the defined dependence tree which imposed the dependence relationships between the variables was selected subjectively, as shown in Fig. 2. Then the second-order marginal distributions were randomly generated. The procedure by which these were generated was as follows: If we define the entire space of each variable to be between 1 and 1000, this space is sub-divided into a number of subspaces with equal width, say 100. That means that we limit ourselves to be dealing with 10 events where each event represents a sub-interval of width equal to 100 from the entire domain as follows: $\{I_1 = [1, 100], I_2 = [101, 200], \ldots, I_{10} = [901, 1000]\}$, thus, effectively simulating a multinomial distribution. In the latter, each outcome is a random number belonging to exactly one of the 10 sub-intervals, I_j, with probability, P_j, where $j = 1, 2, \ldots, 10$. If n_j represents the number of occurrences of values belonging to I_j and n represents the number of independent records, we have

$$\sum_{i=1}^{10} n_i = n, \sum_{i=1}^{10} P_i = 1, \tag{6}$$

where the probability mass function of the multinomial distribution is

$$f(n_1, n_2, \ldots, n_{10}) = \frac{n!}{n_1!\ n_2!\ \ldots\ n_{10}!} \prod_{i=1}^{10} P_i^{n_i}. \tag{7}$$

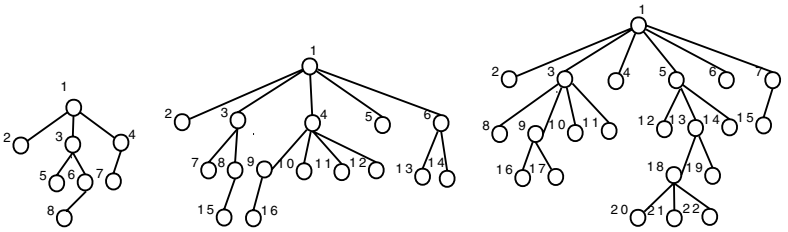
Prior to assigning the second order marginal distributions for the rest of the tree, we had to also randomly generate 10 different probabilities for the most independent variables (*i.e.* to be the root variable).

To randomly populate the file, we can now randomly assign values to the conditional probability from the joint and marginal distributions as follows: If, as per the assumed tree-based dependence, variable V_m, depends on variable V_n, this means we have to define a set of probabilities, $\{P_{nm}\}$, when the value of V_n, say v_{in}, belongs to any defined sub-interval I_j given that the value of variable V_m, say v_{im} belongs to any sub-interval I_l. Thus, $P_{mn} = Pr(v_{in} \in I_j | \ v_{im} \in I_l)$, where i represents the index of the record in the micro-data file and assumes values in $\{1, 2, \ldots, n\}$. The indices j and l represent the indices of the sub-interval where the random variable falls, and which are the result of dividing the entire domain into 10 sub-intervals. Finally, the indices n and m represent the specific dimensions in the micro-data file, and are in the range $\{1, \ldots, d\}$, $n \neq m$.

The above procedure was implemented for all pairwise combinations of random variables associated with the micro-data file.

3.2 Results

The experiments conducted were of three categories: In the first set of experiments the intention was primarily focused on testing whether the best dependence tree can be learned from the continuous micro-data file, and if it sufficiently reflected the dependence model. In the second set of experiments, the goal was



(a) Sim_1 (b) Sim_2 (c) Sim_3

Fig. 2. The true structures for the simulated data sets

primarily to validate our strategy for determining the subset of variables to micro-aggregate the micro-data file, and to study its effect on the value of the IL. The last set of experiments considers the case when we work with continuous vectors in which we impose the assumption of Normality to calculate the edge weights.

Experiment Sets 1

The first set of experiments was done on simulated data sets with a known structure of the best dependence tree which is to be inferred by the learning algorithm[3]. To do this, we first scanned the micro-data file to specify the domain space of each variable in the file, and then divided it into a number of sub-intervals sharing the same width. After that, we achieved a categorization phase by replacing the values belonging to a certain sub-interval in each variable by the corresponding category/code. For example, in the case of the simulated data sets, all the variables shared the same domain space between 1 and 1000, which was divided into 10 subintervals, as explained earlier. Consequently, all values belong to the $[1, 100]$ interval were replaced by 1, all values belong to the $[101, 200]$ interval were replaced by 2 and so on. The above procedure was repeated for all the variables so as to generate the categorical micro-data file.

Our results clearly show that the "width" parameter plays a predominant role in controlling the degree of smoothing and estimating the best dependence tree. Our experiments indicated that assigning a suitable value to the width parameter guaranteed the convergence of the MST to the true underlying structure of the best dependence tree. Generally speaking, the value of the width parameter should be large enough to generate a sufficient number of sub-intervals from the defined domain space to guarantee a satisfactory level of smoothing. The actual value used is specified in the respective experimental results.

Consider the tree structure given by Sim_1, Sim_2, and Sim_3 as given in Fig. 2. Approximating the dependence information of the simulated data sets based on the structure of the MST obtained using the $EMIM$ metric succeeded in locating the real structure when the width parameter was set to the values 50,

[3] The peculiarities of the learning and estimation problems are explained in the unabridged version of this paper [10].

100, and 150 for Sim_1, 70, 100, and 120 for Sim_2, and 90, 100 and 110 for Sim_3.

Experiment Sets 2
The second set of experiments verified our conjecture that it was expedient to use the sub-set of the variables obtained (from the best dependence tree) by projecting the micro-data file into 3, 4 or 5 variables before invoking the multi-variate micro-aggregation process.

Since an MAT seeks to reduce the loss in the data utility, it must be pointed out here that the value of the IL depends on the sub-set of variables used to micro-aggregate the multi-variate data file. As mentioned earlier, to infer the best sub-set of variables to be used in the micro-aggregation, we have to go through all the different projection possibilities. The results (Table 1) show that the estimation of the percentage value of the IL for various data sets obtained by projecting the entire data set into specified number of variables prior to invoking the $MDAV$ method. The value of the IL was bounded between the minimum value (in the fourth column) that was obtained by using the variable indices addressed in the third column, and the maximum value (in the sixth column) that was obtained by using the indices addressed in the fifth column. The last column in Table 1 represents the average value of the IL over all the different combinations of projected variables in the micro-data file.

Practically, due to the exponential number of combinations, we could not cover the entire solution space so as to reach to the best sub-set of the variables to be used in the micro-aggregation[4]. As opposed to this, by involving only the vertices that have the maximum number of I/O edges in the connected undirected graph to micro-aggregate the micro-data file, we were able to obtain an acceptable value of the IL close to its lower bound, and which is always (in all the cases) superior to the average value. Thus, such an automated strategy for projecting the multi-variate data sets will reduce the solution space to be searched which, in turn, reduces the computation time required to test the candidate variables, and to choose the best sub-set from them.

Table 2 shows the percentage value of the IL obtained by using our strategy in projecting the micro-data file into sub-sets of sizes 3 and 4, respectively, prior to invoking the $MDAV$ method. When the simulated sets were projected onto three variables prior to the micro-aggregation, the minimum values of the IL were equal 38.11% for $Sim1$, 51.95% for $Sim2$ and 55.82% for $Sim3$. These values were quite close to the lower bound of the IL which were equal to 37.64% for $Sim1$, 51.65% for $Sim2$ and 55.52% for $Sim3$, respectively.

Experiment Sets 3
The distribution of the average of a set of random variables tends to be Normal, even when the distribution of the individual random variables is decidedly non-Normal. This is a consequence of the Central Limit Theorem, which is the foundation for many statistical procedures, because the distribution of the

[4] On our processor, it took up to a few hours or even days depending on the dimensionality and cardinality of the data set, to exhaustively search the entire space.

Table 1. The value of the IL obtained by using the $MDAV$ multi-variate MAT after projecting various data sets into the specific number of variables

Data Set	No. of projected variables	No. of possible combinations	Indices of the variables used to obtain the min. value of IL	The min. value of IL	Indices of the variables used to obtain the max. value of IL	The max. value of IL	The average value of IL
Sim_1	1	8	2	56.6216	3	59.8342	58.6944
	2	28	1,8	46.8576	3,5	51.3179	49.4823
	3	56	2,4,7	37.6486	4,6,7	42.2961	39.7095
	4	70	2,4,7,8	37.6486	4,6,7,8	42.2961	39.5901
	5	56	1,3,5,6,7	37.6522	3,4,6,7,8	42.1577	39.7102
Sim_2	1	16	2	61.6118	11	63.0976	62.7308
	2	120	1,16	56.6698	9,13	59.0037	58.2026
	3	560	1,8,11	51.6551	6,8,9	54.3367	53.3249
	4	1820	1,8,11,12	51.6551	6,8,9,10	54.3367	53.18
Sim_3	1	22	2	62.5849	7	64.0993	63.7251
	2	231	2,22	59.0211	8,11	60.9375	60.4842
	3	1540	2,7,13	55.5274	4,6,14	57.7998	56.9827

Table 2. The value of the IL obtained by using the $MDAV$ multi-variate MAT after projecting various data sets using 3 or 4 variables by using the $EMIM$ metric to calculate the edge weights in the connected undirected graph

Data set	k value	Width value	No. of possibilities	Variable indices	IL
Sim_1	3	100	2	1,3,4 1,3,6	51.9684 52.1126
Sim_2	3	100	2	1,3,4 1,3,6	51.9684 52.1126
Sim_3	3	100	2	1,3,5 1,3,18	56.1318 55.8246
Sim_1	4	100	1	1,3,4,6	38.1105
Sim_2	4	100	1	1,3,4,6	51.9684
Sim_3	4	100	1	1,3,5,18	56.1318

phenomenon under study does not necessarily have to be Normal. Therefore, the last set of experiments assumes the Normality of the micro-data file to quickly compute the first and second order marginals, and to thus lead to the MST for computing the best dependence tree. Subsequently, we applied our strategy to choose the subset of random variables to project the file before invoking the $MDAV$ method.

Under the assumption of normality, the edge weight, w_{ij}, between two variables V_i and V_j in the connected undirected graph can be calculated by [16]:

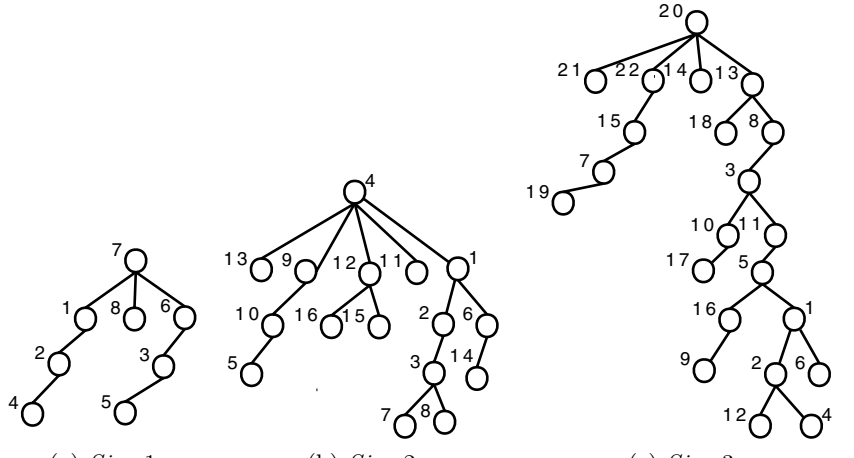
$$w_{ij} = -\frac{1}{2}\, log(1 - \rho_{ij}^2) \tag{8}$$

where ρ_{ij}^2 represents the correlation coefficient between the two variables V_i and V_j in the graph.

The beauty of estimating the dependence model assuming normality is that it does not depend on any parametric value. Therefore, it leads to a unique MST if the edges weight are unique. Figure 3 shows the best dependence tree for the

Table 3. The value of the IL obtained by using the $MDAV$ multi-variate MAT after projecting various data sets into 3 variables assuming normality

Data set	No. of possibilities	Variable indices	IL
Sim_1	6	7,2,3	37.8147
		7,1,3	37.8676
		7,1,6	38.1610
		7,2,6	38.1704
		7,3,6	41.4843
		7,1,2	42.0634
Sim_2	3	4,1,3	51.9684
		4,1,12	52.0159
		4,3,12	53.9267
Sim_3	10	20,1,5	55.6419
		20,2,13	55.6631
		20,2,5	55.8020
		20,2,3	55.8722
		20,1,3	55.9010
		20,1,13	55.9474
		20,5,13	57.1943
		20,3,5	57.3903
		20,3,13	57.4192
		20,1,2	57.4770



(a) Sim_1 (b) Sim_2 (c) Sim_3

Fig. 3. The best dependence tree for the simulated data sets assuming normality

simulated data sets. It is worth mentioning that using the correlation between two random variables in calculating the edges weights of the graph does not lead to convergence to the "true" underlying dependence model in the case of the

simulated data sets. However, generally the overall process yielded a value of IL close to the minimum value of the IL after projecting the entire data set into 3 variables although the search space was greater than the search space that resulted from using the $EMIM$ metrics. The minimum value of the IL was equal to 37.8% for $Sim1$, 51.96% for $Sim2$, and 55.64% for $Sim3$.

4 Conclusions

In this paper, we have shown how the information about the structure of the dependence between the variables in the micro-data file can be used as a fundamental indicator before invoking any MAT. By using this information, we have proposed a new automated scheme as a pre-processing phase to determine the number and the identity of the variables that are to be used to micro-aggregate the micro-data file for minimizing the IL in secure statistical databases. This is achieved by constructing a connected undirected graph whose nodes represent the random variables in the micro-data file, edges represent the statistically dependencies, and the edges weights are computed either by using the $EMIM$ metric or the correlation values when a normality constraint is assumed. The experimental results show that such a methodology involving projecting the multivariate data sets reduces the solution space, which further directly reduces the computation time required to search the entire space combinatorially. In spite of this, this methodology leads to a solution whose IL values are close to the minimum value of the IL that can be obtained by exhaustively searching over the entire search space. Thus, in conclusion, our work has demonstrated the intractability of the MAP and presented a promising tool for enhancing the data utility.

References

1. Adam, N., Wortmann, J.: Security-Control Methods for Statistical Databases: A Comparative Study. ACM Computing Surveys 21(4), 515–556 (1989)
2. Chow, C., Liu, C.: Approximating Discrete Probability Distributions with Dependence Trees. IEEE Trans. Information Theory 14(11), 462–467 (1968)
3. Domingo-Ferrer, J., Mateo-Sanz, J.: Practical Data-Oriented Microaggregation for Statistical Disclosure Control. IEEE Transactions on Knowledge and Data Engineering 14(1), 189–201 (2002)
4. Domingo-Ferrer, J., Torra, V.: Aggregation Techniques for Statistical confidentiality. In: Aggregation operators: new trends and applications, pp. 260–271. Physica-Verlag, Germany (2002)
5. Fayyoumi, E., Oommen, B.: A Fixed Structure Learning Automaton Micro-Aggregation Technique for Secure Statistical Databases. In: Privacy Statistical Databases, Rome, Italy, pp. 114–128 (2006)
6. Hansen, S., Mukherjee, S.: A Polynomial Algorithm for Univariate Optimal Microaggregation. IEEE Transactions on Knowledge and Data Engineering 15(4), 1043–1044 (2003)

7. Laszlo, M., Mukherjee, S.: Minimum Spanning Tree Partitioning Algorithm for Microaggregation. IEEE Transactions on Knowledge and Data Engineering 17(7), 902–911 (2005)
8. Oganian, A., Domingo-Ferrer, J.: On The Complexity of Optimal Microaggregation for Statistical Disclosure Control. Statistical Journal of the United Nations Economic Comission for Europe 18(4), 345–354 (2001)
9. Oommen, B., Fayyoumi, E.: A Novel Method for Micro-Aggregation in Secure Statistical Databases Using Association and Interaction. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 126–140. Springer, Heidelberg (2007)
10. Oommen, B.J., Fayyoumi, E.: On Utilizing Dependence-Based Information to Enhance Micro-Aggregation for Secure Statistical Databases. Unabridged Version of This Paper (to be submitted)
11. Sanchez, J., Urrutia, J., Ripoll, E.: Trade-Off between Disclosure Risk and Information Loss Using Multivariate Microaggregation: A Case Study on Business Data. In: Domingo-Ferrer, J., Torra, V. (eds.) PSD 2004. LNCS, vol. 3050, pp. 307–322. Springer, Heidelberg (2004)
12. Solanas, A., Martínez-Ballesté, A.: V-MDAV: A Multivariate Microaggregation With Variable Group Size. In: 17th COMPSTAT Symposium of the IASC, Rome (2006)
13. Solanas, A., Martinez-Balleste, A., Mateo-Sanz, J., Domingo-Ferrer, J.: Multivariate Microaggregation Based Genetic Algorithms. In: 3rd International IEEE Conference on Intelligent Systems, pp. 65–70 (2006)
14. Torra, V., Domingo-Ferrer, J.: Towards Fuzzy C-Means Based Microaggregation. In: Grzegorzewski, P., Hryniewicz, O., Gil, M. (eds.) Advances in Soft Computing: Soft Methods in Probability, Statistics and Data Analysis, pp. 289–294. Physica-Verlag, Germany (2002)
15. Valiveti, R., Oommen, B.: On Using the Chi-Squared Metric for Determining Stochastic Dependence. Pattern Recognition 25(11), 1389–1400 (1992)
16. Valiveti, R., Oommen, B.: Determining Stochastic Dependence for Normally Distributed Vectors Using the Chi-squared Metric. Pattern Recognition 26(6), 975–987 (1993)

Additive Regression Applied to a Large-Scale Collaborative Filtering Problem

Eibe Frank[1] and Mark Hall[2]

[1] Department of Computer Science, University of Waikato, Hamilton, New Zealand
eibe@cs.waikato.ac.nz
[2] Pentaho Corporation, 5950 Hazeltine National Drive, Suite 340, Orlando, FL, USA
mhall@pentaho.org

Abstract. The much-publicized Netflix competition has put the spotlight on the application domain of collaborative filtering and has sparked interest in machine learning algorithms that can be applied to this sort of problem. The demanding nature of the Netflix data has lead to some interesting and ingenious modifications to standard learning methods in the name of efficiency and speed. There are three basic methods that have been applied in most approaches to the Netflix problem so far: stand-alone neighborhood-based methods, latent factor models based on singular-value decomposition, and ensembles consisting of variations of these techniques. In this paper we investigate the application of forward stage-wise additive modeling to the Netflix problem, using two regression schemes as base learners: ensembles of weighted simple linear regressors and k-means clustering—the latter being interpreted as a tool for multivariate regression in this context. Experimental results show that our methods produce competitive results.

1 Introduction

Collaborative filtering is a challenging application domain for machine learning algorithms, which has gained prominence due to popular web-based recommendation services. The Netflix competition represents a particularly interesting instance of a collaborative filtering problem, in the form of a large movie recommendation dataset that consists of actual movie ratings generated in production use by real users.

The Netflix data is a very demanding problem for prediction methods, due to its size and sparsity. It consists of movie ratings for 17,770 movies, with 100,480,507 ratings in total. The ratings have been provided by 480,189 users. There are approximately 209 ratings for each user on average, so the data is very sparse. It can be viewed as a very large, sparse matrix with 17,770 columns and 480,189 rows. The task is to predict missing entries in this matrix, which correspond to unknown ratings. These predictions can then be used to provide recommendations to users.

One way of tackling this matrix completion problem is to view it as a regression task where the known ratings for a user are used to predict the missing

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 435–446, 2008.

ratings based on a regression model. This is the approach we investigate in this paper. Although the data can be held in memory in less than 1GB if it is stored in sparse form based on index-value pairs—using the data type `short` for the indices and the data types `float` or `byte` for the actual data values (all movie ratings are in the set $1, 2, 3, 4, 5$)—it is a challenging task to build regression models for it. In this paper we present two regression schemes, both based on forward stage-wise additive modeling (FSAM) [6], that are efficient enough to be applicable to this data. The first method uses FSAM in conjunction with an ensemble of simple linear regression models. In the second method, FSAM is used in conjunction with the k-means clustering method.

The paper is structured as follows. In the next section we describe two variants of the FSAM method, one for uni-variate prediction, and one for multi-variate prediction. In Section 3 we present our ensemble of simple linear regression functions, a heuristic base learner for the uni-variate FSAM scheme, and give some results obtained on the Netflix data. In Section 4 we describe how we applied k-means clustering in conjunction with the multi-variate FSAM scheme and present results. Section 5 briefly describes some related work. Section 6 has some conclusions.

2 Regression Using Forward Stage-Wise Additive Modeling

Forward stage-wise additive modeling (FSAM) [6] is a simple technique for fitting an additive model to a given prediction problem. In the case of classification, it is closely related to the well-known boosting methodology. Here we consider the regression version (see Equations 6 and 7 in [6]).

The output of the FSAM scheme is a collection of prediction models, i.e. an ensemble. In the following we call these individual models "base models". The base models are regression models that predict a numeric target based on a given set of independent variables. In this section we assume that we have suitable algorithms for fitting base models to the data. The corresponding algorithms we used for the Netflix problem are discussed in the next two sections.

The FSAM algorithm for additive regression is a greedy algorithm for fitting an ensemble of base models to a given regression problem. Base models are fit in a stage-wise manner, where each model in the sequence is trained on a transformed target consisting of the residual errors that remain from the models built so far. It is assumed that there is an implicit 0th model in the sequence that simply predicts the mean target value observed in the training data. Figure 1 shows the FSAM process based on hypothetical target data and hypothetical predictions obtained from the base models. Once an ensemble of base models has been generated based on a certain number of iterations, ensemble predictions are formed by simply adding the predictions of the base models in the sequence.

Assuming the algorithm used for building the base models minimizes the squared error of the respective residual-based predictions problems, the FSAM

Target values (ratings for a movie)	3.0	2.0	4.0	3.0	
Predictions of Model 0 (mean)	3.0	3.0	3.0	3.0	
Residuals	0.0	-1.0	1.0	0.0	SSE = 2.0
Hypothetical predictions of Model 1	0.1	-0.6	0.4	-0.1	
Residuals	-0.1	-0.4	0.6	0.1	SSE = 0.54
Hypothetical predictions of Model 2	-0.05	-0.1	0.2	0.2	
Residuals	-0.05	-0.3	0.4	-0.1	SSE = 0.2625

Fig. 1. Illustration of the uni-variate FSAM process with hypothetical target values and hypothetical predictions from base models; the sum of squared errors is also given. Note that the values in the first row would correspond to a column in the data matrix.

algorithm for regression greedily minimizes the squared prediction error of the ensemble as a whole.

The algorithm described so far predicts a single target based on some independent variables. It can be applied to the Netflix data on a per-movie basis, treating each movie individually as a target. However, the Netflix problem is essentially a matrix completion problem, where we would like to predict all missing entries for a user based on the existing entries, and we would like to predict them simultaneously because this would be computationally much more efficient. Fortunately it is straightforward to adapt the FSAM method to model multiple targets simultaneously, assuming the base learner is able to also do so: rather than working based on residuals of a single target variable only, we consider the residuals of all variables simultaneously. In each iteration of the FSAM method, a (sparse) matrix consisting of residual errors is passed to the base learner, which then builds a base model that approximates this matrix. The resulting predicted residuals are then used to compute residuals for the next iteration, and so on. Figure 2 shows the multi-variate FSAM process based on a hypothetical sparse data matrix and hypothetical predictions obtained from the base models. At prediction time, the predictions of the base models are again simply added together. In Section 4 we show how the k-means clustering algorithm can be used as the base learner in this multi-variate scheme.

As with many other learning algorithms, the FSAM method can suffer from overfitting. In practice, on the Netflix data, it is generally the case that the error of the FSAM ensemble decreases at first as more iterations are performed (i.e. more ensemble members are used), but it starts to increase after a certain point. A standard trick to delay the point of overfitting, thus making it possible to perform more iterations and potentially build a more accurate ensemble, is to apply shrinkage to the predicted residuals [6]. In this case, the predictions are multiplied by a shrinkage parameter with a value in $(0, 1]$ (see Equation 7 in

Target values (4 movies, 3 users)

3.0		2.0	5.0
	3.0	1.0	1.0
5.0	3.0	3.0	

Predictions of Model 0 (mean)

4.0		2.0	3.0
	3.0	2.0	3.0
4.0	3.0	2.0	

Residuals

-1.0		0.0	2.0	SSE = 12.0
	0.0	-1.0	-2.0	
1.0	0.0	1.0		

Hypothetical predictions of Model 1

-0.7		0.1	1.9
	-0.1	-0.8	-1.8
0.5	-0.2	0.9	

Residuals

-0.3		-0.1	0.1	SSE = 0.5
	0.1	-0.2	-0.2	
0.5	0.2	0.1		

Hypothetical predictions of Model 2

-0.2		0.1	0.1
	0.1	-0.1	0.1
0.6	0.2	-0.1	

Residuals

-0.1		-0.2	0.0	SSE = 0.2
	0.0	-0.1	-0.3	
-0.1	0.0	0.2		

Fig. 2. Illustration of the multi-variate FSAM process with a sparse matrix consisting of hypothetical target values for three users and four movies, and hypothetical predictions from base models; the sum of squared errors is also given

discussion section of [6]) before being used to compute new residuals for the base learner to approximate. We will investigate the effect of the shrinkage parameter in our experiments with k-means in Section 4.

3 Ensembles of Weighted Simple Linear Regressors

The Netflix data is very high-dimensional: there are 17,770 possible movie ratings. This means it is desirable to apply a base learning algorithm in the FSAM method that is computationally efficient in the number of attributes. Another feature of this data is the large number of missing values: on average, the 480,189 users in this data have rated only about 209 movies. This means it is essential to exploit data redundancy at prediction time, making use of "redundant" attributes (i.e. columns in the matrix that correspond to different movies). This section presents a very simple and fast heuristic base learner that we have used successfully to apply the uni-variate FSAM method to the Netflix data.

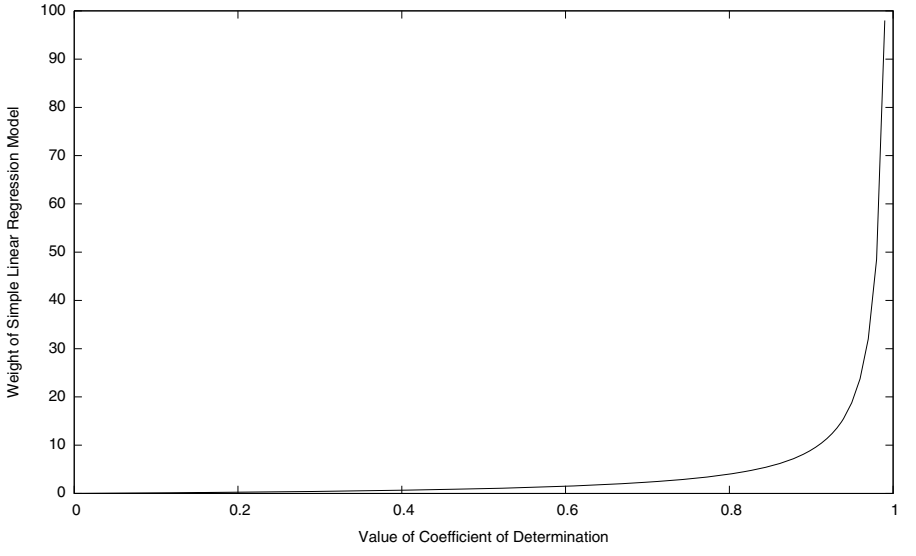


Fig. 3. Function used to compute weight of one simple linear regression model

The basic idea is to to build a single simple linear regression model for each of the movies in the data, excluding the movie for which we want to predict ratings, to form an ensemble of (17,770 - 1) predictors. The simple linear regression models can be built very quickly: computing the sufficient statistics for a simple linear regression model requires a single pass through the (sparse) data.

Based on this ensemble of simple linear predictors it is then necessary to form an overall prediction for a new test case—a target movie rating for a particular user. Obviously only those linear regression models can be used to generate this prediction for which ratings for the user concerned are available in the training data. The most straightforward strategy is to simply average the predictions of the corresponding linear regression models. However, a straight unweighted average does not perform very well. Instead, we found that a simple weighting strategy produced much better results: each linear regression model's prediction is weighted based on the coefficient of determination of the linear model (i.e. the square of the correlation coefficient). It can be computed based on the same sufficient statistics as the linear regression model.

Assuming a value R^2 for the coefficient of determination, we compute the weight of a particular model as as $R^2/(1-R^2)$. This means that more predictive linear regression functions get a higher weight in the ensemble. The weight as a function of R^2 is shown in Figure 3. Using just R^2 as the weight did not appear to give a sufficiently high weight to the most accurate models among the different movie predictors. Hence we adopted this non-linear transformation function.

We also found that it is beneficial to discard simple linear regression models from the ensemble for which we have less than a certain number of movie ratings

in the training data (i.e. models that do not have sufficient support in the training data). For the results in this paper we discarded all models that were based on less than 50 training instances.

Moreover, significantly improved performance was obtained by a simple data normalization step that is fairly standard for the Netflix data, namely by "centering" the input data for each user in the following way: for each user, a smoothed mean of the movie ratings for that user was subtracted from each rating for that user, and the maximum rating possible (the value 5) was added to the result. Assuming the user mean was μ_u, and n_u movie ratings were given for that user in the training data, the smoothed user mean was computed as: $(1 - 1/(n_u - 1)) \times \mu_u + (1/(n_u - 1)) \times 3$. At prediction time, the smoothed mean for the corresponding user was added to the prediction obtained from the model and the value 5 subtracted from this. Resulting values outside the [1,5] interval were clipped. All the results that are stated in this section are based on data normalized in this fashion.

The resulting ensemble of uni-variate predictors gives reasonable performance. It produces a root mean squared error (RMSE) of 0.955 on the Netflix probe set when the probe set is eliminated from the full training set so that an unbiased performance estimate can be obtained.[1] However, we can plug it into the uni-variate FSAM method from the previous section to obtain improved performance. In this case, the ensemble of simple linear regression models is not applied to model the values of the target movie directly; instead, it is used to model the residual errors in the target predictions computed in the forward stage-wise additive modeling strategy. Applying this method to predicting the probe data, by building a model for each of the 17,700 possible target movies using 5 iterations of additive modeling, results in an RMSE of 0.924 on the probe data. This is in the same ball park as results obtained using variants of singular value decomposition on this data (see Section 5 for references).

Figure 4 shows the RMSE on the probe set as the number of FSAM iterations is increased. These results were obtained without any shrinkage applied to the residuals in the FSAM method. We can see that after a certain number of iterations performance starts to deteriorate, indicating that overfitting occurs. It is possible to delay the point at which the error starts to increase by performing shrinkage, leading to further small reductions in error. We investigate the effect of shrinkage in more detail for the multi-variate case in the next section.

4 Modeling Multiple Targets Simultaneously with *k*-Means

A major drawback of the modeling strategy described in the previous section is that it requires building a separate additive model for each of the possible target movies. Given the large number of possible targets this is a time-consuming

[1] The probe set specifies a validation set for models built for the Netflix problem and consists of 1,408,395 ratings from the full dataset, leaving 99,072,112 ratings for training.

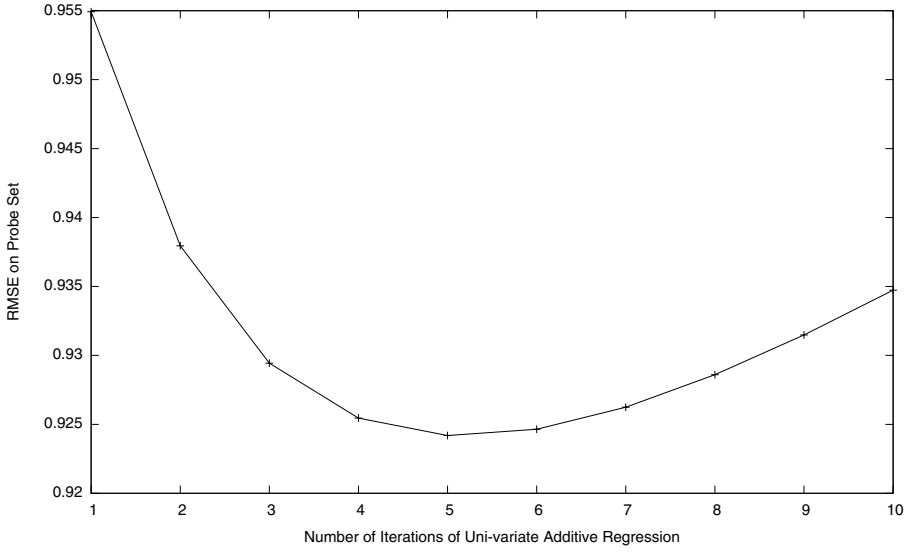


Fig. 4. Root mean squared error on probe set for uni-variate additive regression applied to ensembles of weighted simple linear regression models

process. A much more efficient approach is to model all targets simultaneously by building a multi-variate model, a model that can be used to predict an arbitrary element in the data matrix. In the context of forward stage-wise additive modeling this only requires us to adopt an appropriate multi-variate base learner: as described in Section 2, it is trivial to modify the stage-wise modeling process to work with multiple residuals.

It turns out that there is a well-known modeling technique that slots nicely into this multi-variate version of the FSAM method, namely k-means clustering. It is well-known that k-means clustering finds cluster centers that represent a local minimum of the sum of squared differences to the cluster centers over all training examples. This is exactly what is required from the base learner in the FSAM method: we can simply cluster users (i.e. rows in the data matrix) according to the residual errors in the movie ratings obtained in the FSAM method (i.e. k-means clustering is applied to the matrix of residuals remaining in a particular FSAM iteration). To compute new residuals, a user is assigned to its nearest cluster centroid based on the residuals that remain from the predictions of the previous ensemble members, and the values stored in that centroid are used as predictions. Figure 5 demonstrates this process using an artificial dataset with 143 instances and two continuous-valued attributes.[2] At prediction time, missing residuals for a user are filled in based on the values stored in the centroid that it is assigned to.

[2] The process is the same if the attributes are discrete-valued movie ratings.

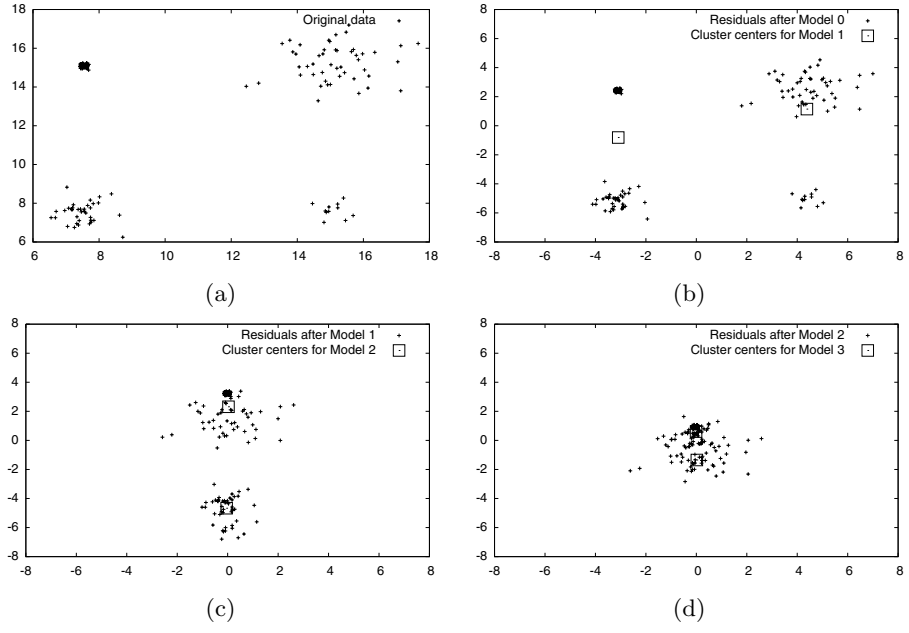


Fig. 5. Multi-variate additive regression with k-means (k=2) based on an artificial dataset: (a) original data; (b)-(d) residuals remaining after iterations 0, 1, 2 respectively, and k-means models obtained from those residuals. (Note: Model 0 predicts the mean of the two attributes.).

The Netflix data is sparse, with many missing values, and the basic method needs to be adapted to deal with this. Fortunately, this is simple: missing values are simply skipped in the distance calculation and in the process of computing the mean residuals stored in the cluster centroids. This also makes the process very efficient due to the sparse representation of the data that is used.

We found that basic k-means produced good results when used in this fashion. However, small improvements can be obtained by applying a global smoothing strategy when computing the cluster centroids: instead of using the per-cluster mean of residuals as the corresponding value for the cluster centroid, we smooth it with the global mean of residuals for the corresponding movie, obtained from all the values for the corresponding movie stored in the matrix of residuals. For the experimental results given below, we used a weight of $1/(1 + n_m)$ for the global mean, where n_m is the number of observed movie ratings for movie m, and one minus this weight for the local mean, when computing the smoothed cluster centroid.

Note that the idea of applying k-means to the Netflix data is not new (see, e.g, [10,9]). However, what we propose here is to use k-means as the base learner in the FSAM method. The k-means clustering algorithm is used to find centroids

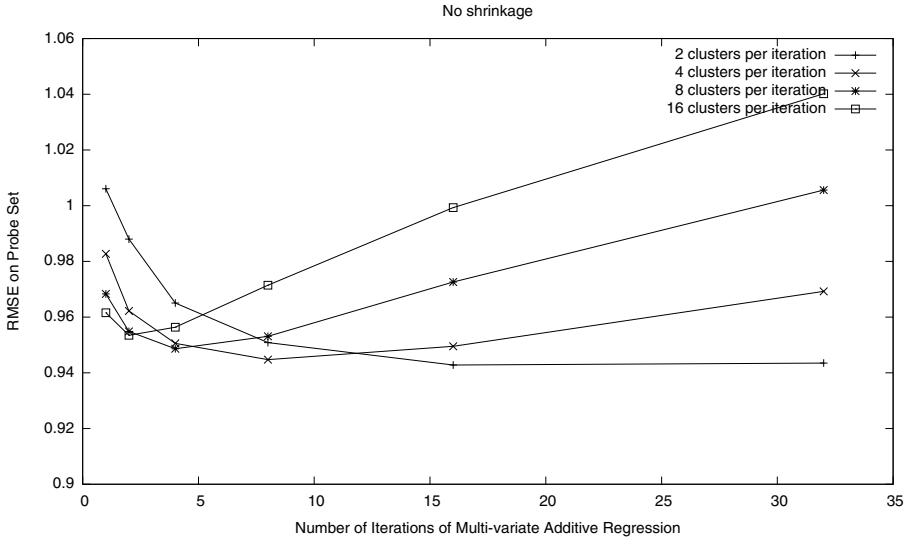


Fig. 6. Root mean squared error on probe set for multi-variate additive regression without shrinkage applied to k-means

of the residuals produced in the FSAM modeling process, and these centroids are also used at prediction time. On the probe set, stand-alone k-means gets an RMSE of 0.962 with $k = 16$ and 0.959 with $k = 32$. We will see that better results can be obtained by using it in conjunction with additive regression.

There are several parameters in the multi-variate FSAM method with k-means: the number of FSAM iterations, the shrinkage value used in the FSAM method, and the number of cluster centers. Another parameter is the number of iterations of the k-means algorithm, but our experiments indicate that this parameter is less important. We fixed it at 40 iterations for the results discussed in the following.

Before presenting the actual RMSE values on the probe set we obtained, let us summarize our findings. We found that it is possible to get good results for different values of k when using k-means in conjunction with the FSAM method. Generally speaking, the larger the value of k, the fewer additive modeling iterations where required to achieve a certain level of performance. However, it was also necessary to adjust the shrinkage value of the additive modeling strategy appropriately. Lower shrinkage values were required for more iterations/cluster centers, to avoid overfitting.

Figure 6 shows the effect of increasing the number of FSAM iterations, for different numbers of clusters (i.e. values of k), keeping the shrinkage parameter fixed at 1 (i.e. no shrinkage). Figure 7 shows the same for a shrinkage value of 0.5, and Figure 8 the results for a shrinkage value of 0.25. The results are

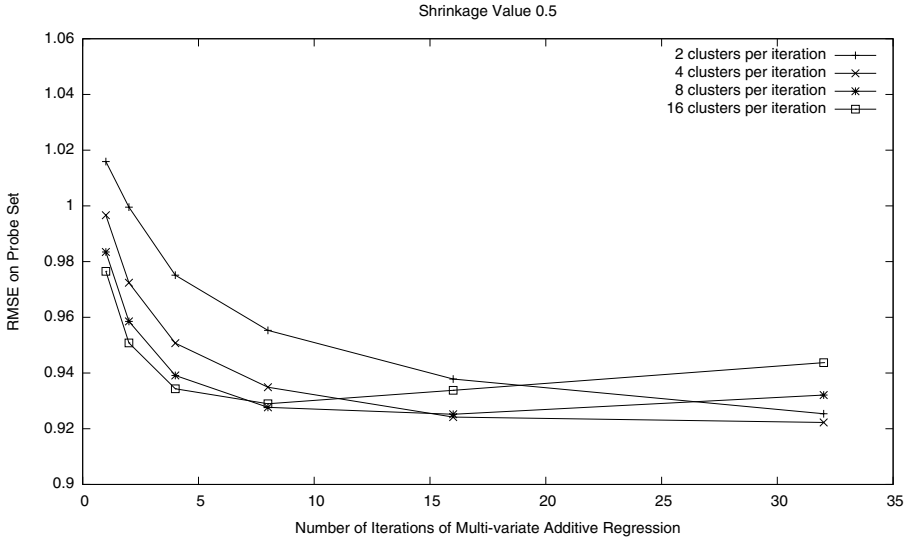


Fig. 7. Root mean squared error on probe set for multi-variate additive regression with shrinkage factor 0.5 applied to k-means

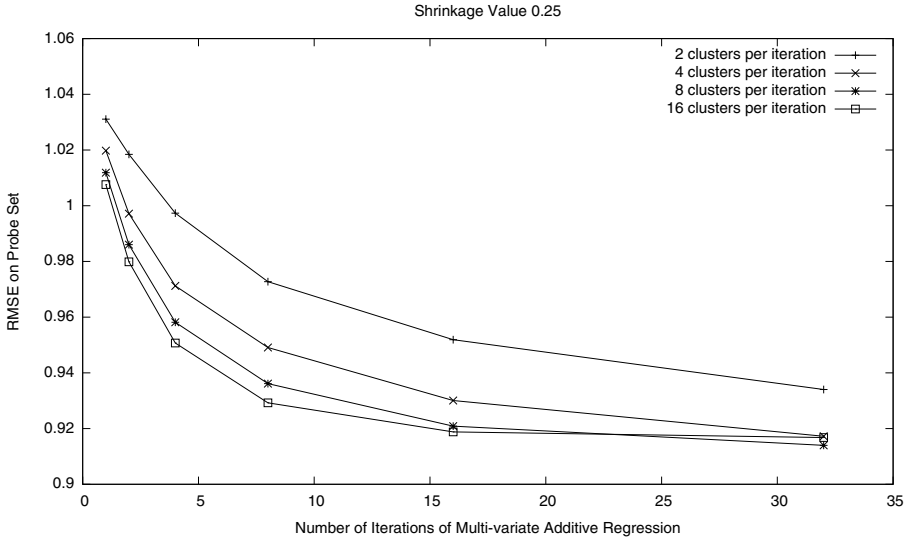


Fig. 8. Root mean squared error on probe set for multi-variate additive regression with shrinkage factor 0.25 applied to k-means

consistent with our above summary. It is beneficial to use more iterations of the FSAM method and/or more cluster centers, provided the value of the shrinkage parameter is adjusted appropriately.

In our experiments we obtained an RMSE value of 0.901 on the probe set by using 4096 iterations of k-means, with k=2, and a shrinkage value of 0.05. Further improvements can be obtained by building multiple additive models using different random number seeds and/or values of k for k-means and averaging their predictions. In this fashion, it is possible to obtain RMSE values below 0.9 on the probe data as well as the actual Netflix test data, which is used in the Netflix competition to evaluate submissions. This appears to be competitive with the best single-paradigm learning schemes that have been evaluated on this data. The best results on the Netflix data so far appear to have been obtained based on large heterogeneous ensembles of diverse learning schemes [1,10,9]. Our schemes could be used as part of such ensembles.

We also experimented with a variant of bisection k-means, an efficient method of applying k-means in a recursive fashion to obtain multiple clusters. In this method, k-means is applied recursively with k=2 to build a tree of clusters in a divide-and-conquer fashion. We investigated this method because it makes it possible to apply a more fine-grained smoothing procedure, where a cluster center is smoothed by blending it with the cluster centers occurring in the path from the root node to its corresponding node in the tree. However, when we used this method in conjunction with additive regression, we did not observe significant increases in performance.

5 Related Work

As mentioned before, k-means has previously been applied to the Netflix data. However, there is also some work on using versions of the FSAM method for this data. Dembczyński et al. [3] investigate ensembles of decision rules for ordinal classification constructed using FSAM. They present two approaches, one based on AdaBoost [4], and the other based on Friedman's gradient boosting machine [5]. The effectiveness of the methods is evaluated on the Netflix data, but only on a small subset of the total number of movies. This makes it difficult to judge how the performance of these methods compares with others on the full Netflix probe set.

Paterek [9] discusses using a linear regression model constructed for each movie. His method differs from the base learner we propose for our first method in three ways: firstly, a multiple linear regression is built for each movie based on binary vectors, that, for each user, indicate the movies that they have rated. Secondly, the prediction for a given rating is adjusted based on a weight proportional to the number of movies the user in question has rated. Finally, the model parameters are learned using gradient descent, which means that the method is relatively slow.

Nearest neighbor methods for collaborative filtering are discussed in [2]. For discussion of mainly SVD-based latent factor models, as applied to the Netflix problem, the reader is referred to [7,8,9,10,11]. Homogeneous and heterogeneous ensembles of neighborhood-based methods and latent factor models are discussed in [9,10,11].

6 Conclusions

This paper has discussed the use of forward stage-wise additive modeling (FSAM) in conjunction with regression schemes for uni-variate and multi-variate prediction on a large-scale collaborative filtering problem, namely the Netflix movie ratings data. Both regression schemes we investigated, ensembles of simple linear regressors for uni-variate prediction and k-means for multi-variate prediction, are fast enough to make FSAM tractable for this application. Results on the Netflix probe set show that both methods achieve good performance. Additive regression with k-means is a particularly attractive scheme because it makes it possible to build a single multi-variate prediction model for the data—effectively a single model that approximates the target matrix and can be used to fill in missing entries in this matrix.

References

1. Bell, R., Koren, Y., Volinsky, C.: Chasing $1,000,000: How we won the Netflix progress prize. ASA Statistical and Computing Graphics Newsletter 18(2), 4–12 (2007)
2. Bell, R.M., Koren, Y.: Improved neighborhood-based collaborative filtering. In: KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)
3. Dembczyński, K., Kotłowski, W., Słowiński, R.: Ordinal classification with decision rules. In: Proc. 3rd International Workshop on Mining Complex Data, pp. 169–181. Springer, Heidelberg (2008)
4. Freund, Y., Shapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
5. Friedman, J.: Greedy function approximation: A gradient boosting machine. Annals of Statistics 29(5), 1189–1232 (2001)
6. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting (with discussion and rejoinder by the authors). Annals of Statistics 28(2), 337–407 (2000)
7. Kurucz, M., Benczúr, A.A., Csalogány, K.: Methods for large scale SVD with missing values. In: KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)
8. Lim, Y.J., Teh, Y.W.: Variational Bayesian approach to movie rating prediction. In: KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)
9. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)
10. Takács, G., Pilászy, I., Németh, B., Tikk, D.: On the Gravity recommendation system. In: KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)
11. Wu, M.: Collaborative filtering via ensembles of matrix factorizations. In: KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)

A Novel Recommending Algorithm Based on Topical PageRank

Liyan Zhang and Chunping Li

School of Software, Tsinghua University
ly-zhang06@mails.tsinghua.edu.cn,
cli@tsinghua.edu.cn

Abstract. In this paper, we propose a Topical PageRank based algorithm for recommender systems, which ranks products by analyzing previous user-item relationships, and recommends top-rank items to potentially interested users. In order to rank all the items for each particular user, we attempt to establish a correlation graph among items, and implement ranking process with our algorithm. We evaluate our algorithm on MovieLens dataset and empirical experiments demonstrate that it outperforms other state-of-the-art recommending algorithms.

1 Introduction

Recommender systems are automatic tools making personalized suggestions to users by analyzing previous interaction information. Generally, they attempt to construct structural models to profile user preference by huge computation, and then recommend the products to potentially interested users. Solutions of recommender systems are mainly classified into three categories: content-based approaches[5], collaborative filtering[7], and hybrid approaches[6]. Recently, some graph based recommending algorithms have been proposed and achieved outstanding prediction performance. As to these methods, some of them rely on the graphs with user-nodes and item-nodes, such as L+, Katz, Dijkstra [1]; others base on the graphs just containing item-nodes, e.g., ItemRank.

ItemRank, proposed by M.Gori and A.Pucci[3], is a random-walk based scoring algorithm. ItemRank refers to Original PageRank, brings the method of item ranking into recommender systems and reports outstanding performance. However, ItemRank still needs some improvements because it fails to take into account item genre and user interest profile, which are important features worth consideration.

In this paper, we propose a Topical PageRank based recommending algorithm, which can be used to rank items for each user and recommend top-ranked items to users, with consideration of item genre and user interest profile. To test the performance, we run a set of experiments on MovieLens dataset, and results show the superiority of our algorithm over existing state-of-the-arts methods.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 447–453, 2008.

2 Algorithm

Generally speaking, the input of a recommender system can be treated as a user-item matrix consisting of tuples, each denoted as $t_{k,i} = (u_k, m_i, r_{ki})$, where u_k is one of users, m_i is one of items, and r_{ki} is an evaluation score, generally an integer ranging from 1 to 5. (Figure 1 is a simple example of a user-item matrix, where "-" means the item was not evaluated by the corresponding user.) Besides that, some information concerning users and items will be provided, e.g., user personal information, item genre information and etc. Our goal is to compute rank score for each item by analyzing the given evaluation scores with respect to each user, and then suggest the top-rank items to users.

	m_1	m_2	m_3	m_4	m_5
u_1	3	5	4	-	-
u_2	4	-	-	-	3
u_3	-	3	-	-	4
u_4	2	-	-	2	-
u_5	4	4	-	-	-
u_6	3	2	5	-	-
u_7	-	-	4	5	-
u_8	4	2	-	4	-
u_9	-	3	-	-	-
u_{10}	-	5	-	2	-

Fig. 1. A simple example of user-item matrix

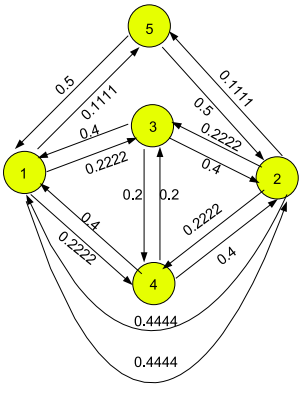


Fig. 2. Example for Correlation Graph

2.1 Data Model: Correlation Graph

A key point for a recommending algorithm is exploiting the item correlations. As an initial step, we first establish a correlation graph among items. Experiences reveal that correlation between items can be indicated by user preference lists, for example, bread and milk are highly associated with each other because plenty of users tend to buy them at the same time. Consequently, we can reasonably assume that item m_i and item m_j are highly related, if they tend to co-occur in preference lists of different users. We define $\mu_{i,j}$ as the set of users u_k who choose both item m_i and item m_j in the training set of user-item tuples Tr (In order to justify the performance of our algorithm, we divide the whole data set into training set Tr and test set Te), that is:

$$\mu_{i,j} = \begin{cases} u_k : (t_{k,i} \in Tr) \wedge (t_{k,j} \in Tr) & \text{if } i \neq j \\ \emptyset & \text{if } i = j \end{cases}$$

Next, we define a $|M| * |M|$ ($|M|$ is the total number of items) matrix $\tilde{\mathcal{M}}$, with each element $\tilde{\mathcal{M}}_{i,j}$ representing the number of users choosing both item m_i

and item m_j. Alternatively, each $\tilde{\mathcal{M}}_{i,j}$ is denoted as $|\mu_{i,j}|$, where $|\ |$ denotes the cardinality of a set. Furthermore, we normalize matrix $\tilde{\mathcal{M}}$ to obtain a stochastic matrix-correlation matrix $\mathcal{M}$, with each element $\mathcal{M}_{i,j} = \tilde{\mathcal{M}}_{i,j}/\omega_j$, where ω_j is the sum of entries in j^{th} column of $\tilde{\mathcal{M}}$. Thus, we obtain a correlation graph $\mathbb{G}$ on basis of the correlation matrix $\mathcal{M}$, in which an edge between item m_i and m_j is established if and only if $\mathcal{M}_{i,j} > 0$.

Taken the data in Figure 1 as an example, the resulting matrix $\tilde{\mathcal{M}}$ is,

$$\tilde{\mathcal{M}} = \begin{pmatrix} 0\,4\,2\,2\,1 \\ 4\,0\,2\,2\,1 \\ 2\,2\,0\,1\,0 \\ 2\,2\,1\,0\,0 \\ 1\,1\,0\,0\,0 \end{pmatrix}$$

Normalizing matrix $\tilde{\mathcal{M}}$, we get the corresponding correlation matrix $\mathcal{M}$. According to matrix $\mathcal{M}$, we could establish the correlation graph $\mathbb{G}$, shown in Figure 2.

2.2 Topical PageRank Based Algorithm

The above item correlation graph in Figure 2 bears two properties: propagation and attenuation, which are two key features for PageRank algorithm. With this observation, we attempt to leverage Topical PageRank algorithm to recommender systems by exploiting genres of items. Specifically, we rank items for different users with Topical PageRank given the same item correlation graph, and recommend top-rank items to potentially interested users.

Firstly, Take a look at Topical PageRank [2], an algorithm for web link-based ranking, which calculates a score vector for each page to distinguish the contribution from different topics. This algorithm is proposed to address the problem of topic drift — a resource that is highly popular for one topic may dominate the results of another topic in which it is less authoritative. Experiments indicate that Topical PageRank outperforms original PageRank.

One of key contributions of Topical PageRank is the introduce of a definition $A_z(p)$, the PageRank score of Page p on topic z, which can be described as follows.

$$A_z(p) = d\alpha \sum_{q:q\to p} \frac{A_z(q)}{O(q)} + d(1-\alpha) \sum_{q:q\to p} \frac{C_z(q)A(q)}{O(q)} + \frac{1-d}{N} C_z(p) \qquad (1)$$

where q denotes any page linked to page p, $O(q)$ represents the degree of page q. $C_z(p)$ is the probability with which page p belongs to topic z, N is the number of pages, d and α are link related and topic related parameters respectively.

Then, we deduce the equivalent matrix form of Equation(1) for different page-topic pairs, defined as follows.

$$A = d\alpha G \bullet A + d(1-\alpha) G \bullet F_{CA} + (1-d)\frac{C}{N} \qquad (2)$$

where G is the normalized connectivity matrix for the corresponding page graph. A and C are page-topic matrix, each element associated with $A_z(p)$ and $C_z(q)$ respectively. F_{CA} is an assistant matrix, $\bullet$ means matrix product.

With the similar idea, we define item rank score matrix for different item-genre pairs as

$$R = d\alpha\mathcal{M} \bullet R + d(1-\alpha)\mathcal{M} \bullet F + (1-d)I \tag{3}$$

where the $|M| * n$ matrix R (M is the total number of items, and n is number of item genres) contains R_{ig} indicating predicted rank score for item m_i on genre g as to a given user. $\mathcal{M}$ is the item correlation matrix obtained above, F is a $|M| * n$ assistant matrix, and I is a $|M| * n$ matrix associated with the original user-item evaluation score. Note that we choose d=0.15, α=0.1 in the following experiments empirically. The iterative formula for calculating R^{u_k} for different users u_k is given by:

$$\begin{cases} R_{ig}^{u_k}(0) = \dfrac{1}{|M| * n}(1 \le i \le |M|, 1 \le g \le n) \\ F_{ig}(t) = (\sum_{g=1}^{n} R_{ig}^{u_k}(t-1)) * P_{ig} \\ R^{u_k}(t) = d\alpha\mathcal{M} \bullet R^{u_k}(t-1) + d(1-\alpha)\mathcal{M} \bullet F(t) + (1-d)I^{u_k} \end{cases} \tag{4}$$

In formula(4), the first equation initializes R^{u_k} and the second one computes assistant matrix F, where P_{ig} refers the probability with which item m_i belongs to genre g, and usually P_{ig} can be calculated directly by the given item-genre information in data set.

Each element of I^{u_k} in the third equation is defined as $I_{ig} = \frac{\tilde{I}_{ig}}{\sum_{g=1}^{n} \tilde{I}_{ig}}$, with $\tilde{I_{ig}} = r_i^{u_k} * P_{ig}$ for user u_k on item m_i, $r_i^{u_k}$ is the initial score that user u_k gave to item m_i. Given totally κ users, we can get κ different I^{u_k}, and thus obtain κ corresponding matrix R^{u_k} by iteratively computing formula(4).

After getting matrix R^{u_k} for each user u_k, we introduce a $\kappa * |M|$ matrix TR, where $TR_i^{u_k}$ denotes the estimated item rank score for user u_k to item m_i. TR^{u_k} is defined as

$$TR^{u_k} = R^{u_k} \bullet Prof^{u_k} \tag{5}$$

where $Prof^{u_k}$ refers to user profile for u_k, denoted as $Prof^{u_k} = r^{u_k} \bullet P$. Furthermore, the vector $Prof^{u_k}$ shows the interest of user u_k on different item genre. By formula (5), we could compute the final result TR^{u_k}, an expected ranking score for all items with regard to user u_k.

The higher $TR_i^{u_k}$ is, the more user u_k prefers item m_i to other lower score items. We finally recommend top-rank items in vector TR^{u_k} to user u_k.

2.3 A Simple Example

In order to clarify the above process, we take the user-item matrix in Figure 1 for example, to calculate the vector TR^{u_1} for user u_1, given the item-genre

matrix P as follows. P could be obtained by the item-genre information, usually provided by the original data set, where P_{ig} denotes the probability with which item m_i belongs to genre g.

$$P = \begin{pmatrix} 0.8 & 0 & 0.2 & 0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.3 & 0 & 0.7 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0.9 & 0 & 0.1 \end{pmatrix}$$

Consequently, according to Figure 1, we get the initial rating array for user u_1, that is $r^{u_1} = (3\ 5\ 4\ 0\ 0)$. Then we can calculate matrix I^{u_1} by analyzing r^{u_1} and P, where $I_{ig}^{u_1}$ refers to the initial rank score of item m_i on genre g, with regard to the given user u_1. Next, with the two matrix I^{u_1}, P, and formula (4), we get matrix R^{u_1} with each $R_{ig}^{u_1}$ indicating predicted rank score for item m_i on genre g as to the given user u_1.

$$R^{u_1} = \begin{pmatrix} 18.8781 & 0.0665 & 5.6275 & 1.3977 \\ 19.6503 & 0.0665 & 1.7153 & 17.8190 \\ 9.8453 & 4.4458e-04 & 20.0578 & 0.6965 \\ 1.6179 & 4.4458e-04 & 0.8188 & 0.6549 \\ 0.6702 & 2.2162e-04 & 0.0901 & 0.3264 \end{pmatrix}$$

After obtaining R^{u_1}, the final result TR^{u_1} can be computed by formula (5), with each $TR_i^{u_1}$ denoting the expected rank score of u_1 to item m_i.

$TR^{u_1} = (11.4820,\ 14.1872,\ 10.8329,\ 1.1909,\ 0.4342)$

Analyzing the final result matrix TR^{u_1}, we could find that the movie priority order for u_1 is m_2, m_1, m_3, m_4, m_5. As a result, except the movies watched by u_1, we would recommend movie m_4 to user u_1.

3 Experiments

To evaluated our algorithm, we run a set of experiments on MovieLens dataset which is constructed from the popular MovieLens Site for recommending movies. MovieLens site has more than 50,000 users who have express opinions on over 3,000 movies. The MovieLens dataset is a standard benchmark for recommender system techniques, containing 100,000 ratings for 1,682 movies by 943 users. The dataset comes with 5 predefined splits, each with 80% of ratings as training set Tr and 20% as testing set Te.

3.1 Evaluation

In order to compare our algorithm with other promising graph based approaches, we choose the degree of agreement (DOA) as the performance measure. We first introduce DOA for a specific user. As to a particular user u_k, the whole movie set can be divided into 3 subsets, i.e., movies in training set Tr_{u_k}, testing set Te_{u_k} and unwatched set Nw_{u_k}. Intuitively, movies watched by the user should rank

higher than movies unwatched. That is to say, movies in training set Tr_{u_k} and testing set Te_{u_k} should rank higher than movies in unwatched set Nw_{u_k}. Thus, we define a Boolean function f_{u_k} to represent the comparison result between a pair of movies from different subsets. If the score $TR_i^{u_k}$ $(m_i \in Te_{u_k})$ is higher than $TR_j^{u_k}$ $(m_j \in Nw_{u_k})$, f_{u_k} is set 1, otherwise, f_{u_k} is 0.

$$f_{u_k} = \begin{cases} 1 & \text{if } TR_i^{u_k} \geq TR_j^{u_k} \\ 0 & \text{if } TR_i^{u_k} < TR_j^{u_k} \end{cases}$$

Then we could calculate individual DOA for each user u_k,

$$DOA_{u_k} = \frac{\sum_{(m_i \in Te_{u_k}, m_j \in Nw_{u_k})} f_{u_k}(m_i, m_j)}{|Te_{u_k}| \bullet |Nw_{u_k}|}$$

which represents the percentage of correct orders with regard to total order pairs for a given user. Computing the average of DOA_{u_k} for all users, we could obtain the final performance index DOA.

3.2 Experiment Results

In this experiment, firstly we study how the 2 parameters d and α affect the performance. Figure 3 shows the variance of performance result DOA with different parameter settings on 10000 ratings randomly chosen from the training set of split5. It is demonstrated by Figure 3 that we get better performance when choosing $d = 0.15$ and $\alpha = 0.1$.

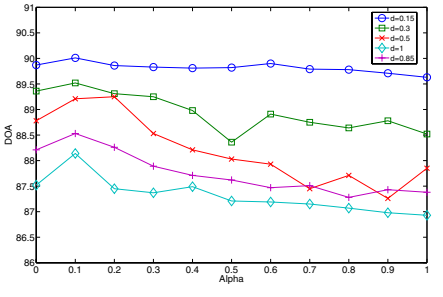


Fig. 3. Variance of Performance with different parameter settings

	Split1	Split2	Split3	Split4	Split5	**Mean**
PR	87.69	87.71	87.65	87.51	88.14	87.74
TR	89.05	89.12	89.26	88.97	89.01	**89.08**

Fig. 4. DOA comparison of our algorithm (TR) with PaperRank (PR) on the 5 splits

Figure 4 demonstrates that the proposed Topical PageRank based algorithm (TR) outperforms ItemRank in all the 5 splits. What improves our experimental results is the process of predicting rank score for each item in each genre and then calculating total rank score for each item. This indicates that the consideration of item genre and user interest profile is important.

Figure 5 shows the superiority of our algorithm (TR), compared with other graph-based algorithms mentioned in[1]. This also proves the advantage of ranking algorithm in recommender systems.

	L+	PR	Katz	Dijkstra	**TR**
DOA	87.18	87.74	85.26	49.65	**89.08**

Fig. 5. Overall DOA comparisons of our algorithm (TR) with other scoring algorithms

4 Conclusions

In this paper, we present a Topical PageRank based algorithm for recommender systems. In this model, we establish an item correlation graph by investigating user-item relationship, rank all the items for each user and recommend the top-ranked item to the corresponding customer. Experiments on MovieLens Dataset show our algorithm outperforms other state-of-the-art graph based algorithms.

Acknowledgement

This work was supported by National Nature Science Funding of China under Grant No. 90718022.

References

1. Fouss, F., Pirotte, A., Saerens, M.: A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. In: IEEE/WIC/ACM International Joint Conference on Web Intelligence, pp. 550–556 (2005)
2. Nie, L., Davison, B.D., Qi, X.: Topical link analysis for web search. In: ACM SIGIR Conference on Information Retrieval, pp. 91–98 (2006)
3. Gori, M., Pucci, A.: ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 2766–2771 (2007)
4. Gori, M., Pucci, A.: Research paper recommender systems: A random-walk based approach. In: IEEE/WIC/ACM International Joint Conference onWeb Intelligence, pp. 778–781 (2006)
5. Adomavicius, G., Tuzhilin, A.: Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. In: IEEE Transactions on Knowledge and Data Engineering, pp. 634–749 (2005)
6. Kim, D., Yum, B.: Collaborative Filtering Based on Iterative Principal Component Analysis. In: Expert Systems with Applications, pp. 823–830 (2005)
7. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In: 29th ACM SIGIR Conference on Information Retrieval, pp. 501–508 (2006)

DynamicWEB: Adapting to Concept Drift and Object Drift in COBWEB

Joel Scanlan, Jacky Hartnett, and Raymond Williams

School of Computing and Information Systems, University of Tasmania, Tasmania, Australia
{joel.scanlan,j.hartnett,r.williams}@utas.edu.au

Abstract. Examining concepts that change over time has been an active area of research within data mining. This paper presents a new method that functions in contexts where concept drift is present, while also allowing for modification of the instances themselves as they change over time. This method is well suited to domains where subjects of interest are sampled multiple times, and where they may migrate from one resultant concept to another due to Object Drift. The method presented here is an extensive modification to the conceptual clustering algorithm COBWEB, and is titled DynamicWEB.

Keywords: Data Mining, Contextual Clustering, Concept Drift.

1 Introduction

Everything changes with time. Ideas change, humans change, concepts change. This is evident in many areas and data mining is no different. The field of concept drift aims to notice changes within a given dataset, and then adapt to these changes. Learning algorithms that allow for concept drift are able to be more robust over time. Within online learning applications being able to adjust a class definition as a result of changes within a domain allows a learning method to produce a model which is accurate up to the current moment.

Concept drift, as examined within the context of data mining, has been studied since the 1980s. Various techniques for detecting and reacting to change within the class descriptions in a dataset have been developed [1, 2]. However, this paper discusses a new method that is closely related to concept drift methods, but is focused upon the changes of individual objects which are examined multiple times over a given time period, where they might drift from one resultant class into that of another. We refer to this as *Object Drift*. As these objects change with time, drift within the concepts which they form can also take place. These two different forms of drift are accounted for by the algorithm described in this paper, whereas traditional concept methods only account for concept drift.

This paper will first examine previous work within the area of concept drift and conceptual clustering algorithms before presenting the algorithm completed by the authors known as DynamicWEB. Some preliminary results will then be discussed.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 454–460, 2008.

2 Conceptual Clustering

Conceptual Clustering as first outlined by Michalski [3] aims to produce concept descriptions for each class. This then allows for clusters to have a simple conceptual interpretation based upon these descriptions. Data clustering methods, while often useful for classification are not as simple to interpret or fully understand from the human perspective. The goal of conceptual clustering extends beyond that of data clustering to not only discover the relationships within the data, but also to discover human interpretable clusters.

This paper presents a method, entitled DynamicWEB, which at its core is a substantial modification to the COBWEB unsupervised incremental conceptual clustering algorithm. This is not the first time that the COBWEB algorithm has been modified to adapt to concept drift, and an existing method will also be briefly examined in this paper.

The COBWEB algorithm was published by Fisher [4] and builds upon the work completed by Michalski [3], and the UNIMEM [5] and CYRUS [6] systems by other authors. While COBWEB draws from these methods, the most significant related work which is also incorporated into COBWEB is that of the Category Utility by Gluck and Corter [7, 8]. The COBWEB algorithm utilises a hierarchical tree to group the observed instances into concepts where traits are shared across the resident instances. The measure COBWEB uses to group the instances together is the category utility.

Gluck and Corter were able to show, using the category utility, similar basic level categorisation to that found within human psychological testing. Basic level categorisation, as used by Gluck and Corter, was first described by Mervis and Rosch [9]. A basic level category is defined as one which is preferred to a more generalised or specific category. Fisher showed that by using the category utility upon the attribute value pairs present within standard data mining datasets a probabilistic conceptual clustering algorithm could be produced. The category utility functioning within COBWEB is sufficiently resilient to produce a useful measure of likeness. Our research does not modify this calculation; for an in-depth explanation of its operation and derivation refer to Gluck and Corter [7, 8] or Fisher [4]. COBWEB's control structure (Table 1) remains as presented by Fisher and Gennari [10], with the modifications we have made being detailed in Section 4.

Table 1. The COBWEB insert mechanism

1.	Search children for the best match to the current instance (start at root for new instance)
2.	Create a new class; calculate category utility
3.	Incorporate into the best matching existing child at this node; calculate category utility
4.	Merge the two best matching children and incorporate forming a new class; calculate category utility.
5.	Split the best matching child; calculate category utility
6.	Select the option from above with the greatest category utility and then continue at 1. If the category utility is below the cut-off then move to 7.
7.	Move on to the next instance at step 1, continuing till the end of the dataset.

Since the publication of COBWEB there have been several other authors who have modified the method to further the research. COBBIT, by Kilander and Jannsson [11], is a variation of COBWEB that allows it to adapt to concept drift through the use of a time window. A time window is an approach that has been used by other authors in approaching the problem of concept drift [2] where a set number of instances smaller than the total is used within the tree at any given moment. As time progresses only the most recent instances in that window are utilised. COBBIT is able to adapt to concept drift, but not object drift.

3 DynamicWEB

DynamicWEB was created to allow objects to be tracked over time as they change within their domain of knowledge. Example domains include a person's behaviour being tracked within a security system, the weather at a given location as it changes or a person's health over a time period. It is common within data mining for methods to be able to examine latitudinal datasets, where many objects are sampled once. Patterns are extracted, and knowledge is gained. But there are few methods for longitudinal studies where the same person, or object, is sampled many times requiring models to be updated with the new information. It is this problem space which DynamicWEB is investigating.

Concept drift is a related area to this but instead of just the concept changing with time, DynamicWEB looks at objects that change with time as well. As DynamicWEB aims to follow the individual objects as they change, the overarching concepts are also able to drift. As such, DynamicWEB allows for both concept and object drift. As a given object changes with time the instance within the DynamicWEB tree is located, updated and re-clustered with respect to its neighbours reflecting the new information gained from its most recent change. To fully explain how this was undertaken made the modifications to COBWEB will now be discussed.

COBWEB in its original form was not intended to have updates occur to the instances that were present within the tree structure. The tree is sorted based on the likeness of the objects resident within it and to locate a given instance each instance would have to be examined in turn, resulting in a O(n) search time. Further, there was no provision for removing, or modifying, an instance within the tree once it was placed there. A modification of an instance within the tree itself would change the category utility of the node containing it, and any parent nodes, thus destroying integrity of the tree. Therefore deletion and modification methods are needed along with a faster way of searching the tree to enable these operations to occur in a more efficient manner.

An index for the tree was implemented (Figure 1) using an AVL tree. For each instance that was added to the tree the identifier for the object was also stored within the AVL tree along with a pointer to its location within the clustering tree. The AVL tree could now be searched to locate the instances which have been clustered in a search time of O($log\ n$). When considering the update mechanism (Table 2), and the various scenarios in which an item might require to be updated it was decided that the most appropriate action would be to remove the item from the tree, adjust the surrounding nodes, and then re-add the item. While there are scenarios where simply

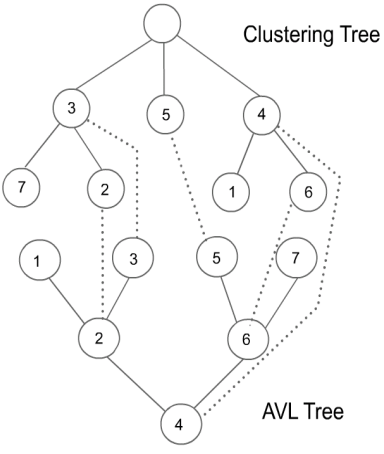


Fig. 1. This represents the way that the AVL tree acts as an index for the clustering tree. The AVL tree is sorted by an identifier, while the clustering tree is sorted by attribute similarity.

Table 2. The DynamicWEB update mechanism

1.	Search AVL tree for the instance to be updated, returning the node it is located at.
2.	Remove the instance from the node in the tree
3.	Update the relational statistics at that node
4.	Consider carrying out an operation on that node I. Is the node empty and needs to be removed II. Should the node be split? III. Should the node be merged? For options II and III calculate the category utility, comparing to the current value.
5.	If the update option produced a category utility that is better than the current, and greater than the cut-off, then perform operation. For all instances affected by an operation update their nodes within the AVL tree.
6.	If the current node is the root, or the cut off is greater than the suggested category utilities move onto step 7. Else move to the parent and perform step 2 onwards.
7.	Update the instance with the new information that has arrived. This may include I. Calculating derived attributes II. Replacing values
8.	Insert updated instance back into the clustering tree, updating the AVL tree location.

updating the instance in place and then flowing those effects through would be acceptable, and indeed in these cases it would be more efficient, this is not true of all cases. In some cases where attributes are replaced and the variation could cause the instance to move to a new location a significant distance away, the migration process would be cumbersome. To avoid this, the update mechanism removes the instance of interest, updating the tree recursively back to the root, and then re-adds the updated instance to the tree. The instance may contain attributes derived from multiple different observations of the object, thus creating a profile across the multiple observations.

As the instance changes across multiple observations it may migrate from one classification to another or may migrate to a better position in the tree.

4 Experiments

DynamicWEB is quite a different method to those discussed previously. Its main strength is being able to update and keep track of individual objects over time as they change. Concept drift algorithms have previously focused on keeping track of a class definition, or measuring the effectiveness of algorithms at recovering a class definition that has been changed drastically. Figure 2 shows a comparison between DynamicWEB and COBBIT [11] using the STAGGER concept dataset [1]. The dataset is a series in which two sudden drifts occur within the resulting classes of the instances. The learning methods are then required to re-learn the resulting class definitions. Both STAGGER and FLORA used this dataset as a method of measuring how quickly a class could be relearned after a sudden drift [1, 2]. COBBIT is using a window size one quarter the size of each concept group (10 instances). COBBIT starts to recover from the sudden drift faster than DynamicWEB, but is overtaken by the midpoint of each concept group.

DynamicWEB can be of use in traditional concept drift problems. However its primary goal is to track when individual objects change over time and drift from one class to another, independent of whether the classes are drifting or not. Figure 3 illustrates a small dataset that the authors created to show the way DynamicWEB functions on larger datasets. The dataset is based upon Quinlan's [12] weather dataset, but is expanded to include 3 measurements of 12 locations. The class values given to the instances were assigned by using Quinlan's C4.5 [13]. Over these three measurements each location's temperature, humidity, rain and wind profiles are updated. Some locations start off within the positive class, and then migrate to the negative; some remain where they are; others migrate in the opposite direction.

To illustrate that DynamicWEB can scale to a much larger dataset a third test was completed on the Physiological Data Modeling Contest dataset [14]. DynamicWEB

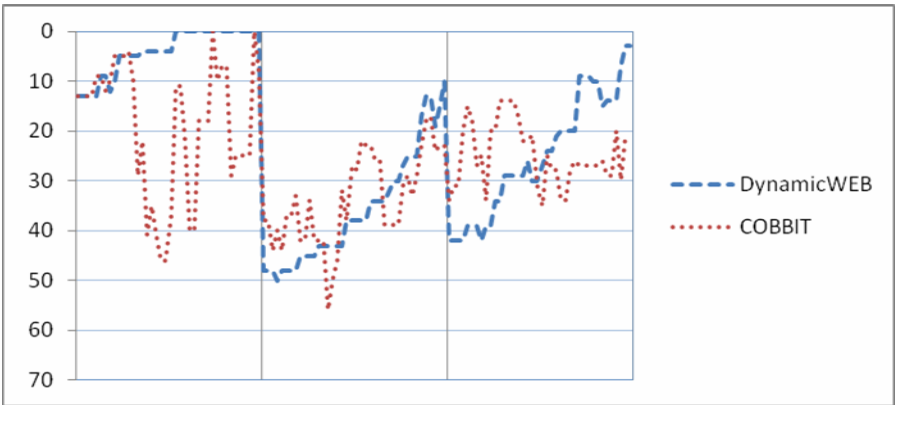


Fig. 2. A comparison of DynamicWEB vs COBBIT with the STAGGER concepts dataset

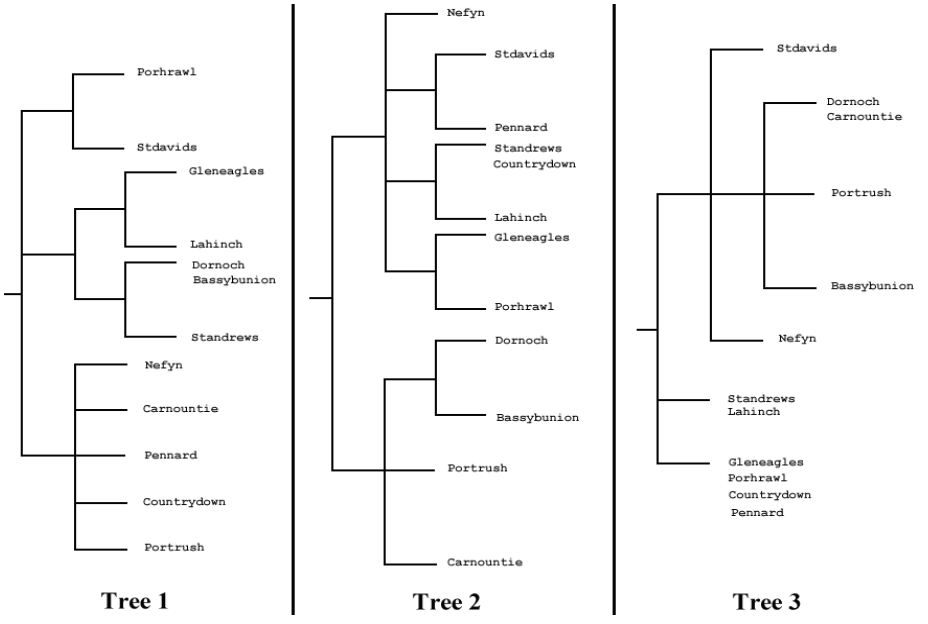


Fig. 3. The progression of the DynamicWEB tree across three updates to a modified weather dataset. Each item within the tree is updated twice resulting in the modifed trees presented.

performed well monitoring the 35 objects, each of which was updated fifteen thousand times within the dataset.

5 Further Work

DynamicWEB is currently being run on more datasets of various sizes, from several domains to illustrate its generality and usefulness across multiple knowledge domains.

DynamicWEB was also designed to operate using multiple clustering trees in parallel. Other authors [1, 2] have discussed the negative impact of noise upon concept drift, and the authors of this paper believe that it may be possible to reduce this impact by parallelising some of the clustering. Within datasets containing many attributes, some of the attributes operate independently of each other, and can act as noise to each other. This effect could be worsened by inherent noise within these attributes. By parallelising the clustering across multiple trees it may be possible to reduce this impact. This feature is currently being tested upon other datasets not shown here, and has previously been discussed elsewhere [15].

6 Conclusion

This paper discussed and presented a new method for adapting to concept drift as well as adapting to object drift that is occurring within objects which are being observed multiple times.

The examination and monitoring of individual objects across multiple observations is a natural and useful extension to the investigation of concept drift. DynamicWEB is presented as a novel approach to this problem space, while still being grounded within proven data mining theory by extending the respected method COBWEB [4]. The method is currently being tested upon different datasets from many domains in an effort to prove both its generality and usefulness in data mining universally.

A comparison between DynamicWEB and COBBIT using the STAGGER concept dataset was presented. Also shown was a diagram illustrating the changes that occurred after each objects' profile within the tree was updated twice.

In further work DynamicWEB will be enabled to perform conceptual clustering in parallel across multiple linked classification trees in an effort to reduce the impact of unwanted interactions between independent variables.

References

1. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Mach. Learning 1(3), 317–354 (1986)
2. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)
3. Michalski, R.S.: Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. International Journal of Policy Analysis and Information Systems 4(3), 219–244 (1980)
4. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Mach. Learn. 2(2), 139–172 (1987)
5. Lebowitz, M.: Concept learning in a rich input domain: Generalization-based memory. Machine learning: An artificial intelligence approach 2 (1986)
6. Kolodner, J.L.: Reconstructive memory: A computer model. Cognitive Science 7(4), 281–328 (1983)
7. Gluck, M., Corter, J.: Information, uncertainty, and the utility of categories. In: Proceedings of the Seventh Annual Conference of the Cognitive Science Society, pp. 283–287. Lawrence Erlbaum Associates, Irvine (1985)
8. Corter, J., Gluck, M.: Explaining basic categories: Feature predictability and information. Psychological Bulletin 111(2), 291–303 (1992)
9. Mervis, C., Rosch, E.: Categorization of natural objects. Annual Review of Psychology 32 (1981)
10. Gennari, J.H., Langley, P., Fisher, D.: Models of incremental concept formation. Artif. Intell. 40(1-3), 11–61 (1989)
11. Kilander, F., Jansson, C.G.: COBBIT - A Control Procedure for COBWEB in the Presence of Concept Drift. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 244–261. Springer, Heidelberg (1993)
12. Quinlan, J.R.: Induction of decision trees. Mach. Learn. 1(1), 81–106 (1986)
13. Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufmann Publishers, San Francisco (1993)
14. Physiological data modeling contest (2004), `http://www.cs.utexas.edu/users/sherstov/pdmc/`
15. Scanlan, J., Hartnett, J., Williams, R.: Dynamic web: Profile correlation using cobweb. In: 19th Australian Joint Conference on Artificial Intelligence. Springer, Hobart (2006)

L-Diversity Based Dynamic Update for Large Time-Evolving Microdata

Xiaoxun Sun[1], Hua Wang[1], and Jiuyong Li[2]

[1] Department of Mathematics & Computing
University of Southern Queensland, QLD, Australia
{sunx,wang}@usq.edu.au
[2] School of Computer and Information Science
University of South Australia, Adelaide, Australia
jiuyong.li@unisa.edu.au

Abstract. Data anonymization techniques based on enhanced privacy principles have been the focus of intense research in the last few years. All existing methods achieving privacy principles assume implicitly that the data objects to be anonymized are given once and fixed, which makes it unsuitable for time evolving data. However, in many applications, the real world data sources are dynamic. In such dynamic environments, the current techniques may suffer from poor data quality and/or vulnerability to inference. In this paper, we investigate the problem of updating large time-evolving microdata based on the sophisticated l-diversity model, in which it requires that every group of indistinguishable records contains at least l distinct sensitive attribute values; thereby the risk of attribute disclosure is kept under $1/l$. We analyze how to maintain the l-diversity against time evolving updating. The experimental results show that the updating technique is very efficient in terms of effectiveness and data quality.

1 Introduction

Many organizations are increasingly publishing microdata (tables that contain unaggregated information about individuals). These tables can include medical, voter registration, census, and customer data. Some of these microdata need to be released, for various purposes, to other parties in a modified form (without the direct identifying information such as SSN, Name, etc.). But even altered this way, these datasets could still present vulnerabilities that can be exploited by intruders, i.e. persons whose goals are to identify specific individuals and to use the confidential information they discover for malicious purposes. The high volume and availability of released datasets together with ever increasing computational power made the protection against those vulnerabilities an increasingly difficult task. To avoid linking attacks, Samarati and Sweeney [11,15] proposed a definition of privacy called k-anonymity. A table satisfies k-anonymity if every record in the table is indistinguishable from at least $k-1$ other records with respect to every set of quasi-identifier attributes; such a table is called a k-anonymous table.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 461–469, 2008.

Due to its conceptual simplicity, numerous algorithms have been proposed for implementing k-anonymity via generalization and suppression. Samarati [11] presents an algorithm that exploits a binary search on the domain generalization hierarchy to find minimal k-anonymous table. Sun *et al.* [12] recently improve his algorithm by integrating the hash-based technique. Bayardo and Agrawal [3] presents an optimal algorithm that starts from a fully generalized table and specializes the dataset in a minimal k-anonymous table, exploiting ad hoc pruning techniques. LeFevre *et al.* [6] describes an algorithm that uses a bottom-up technique and a priori computation. Fung *et al.* [5] present a top-down heuristic to make a table to be released k-anonymous. As to the theoretical results, Meyerson and Williams [9] and Aggarwal *et al.* [1,2] proved the optimal k-anonymity is NP-hard (based on the number of cells and number of attributes that are generalized and suppressed) and describe approximation algorithms for optimal k-anonymity. Sun *et al.* [13] proved that k-anonymity problem is also NP-hard even in the restricted cases, which could imply the results in [1,2,9] as well.

Recent studies shows that although k-anonymity protects against identity disclosure, it is insufficient to prevent attribute disclosure. To address this limitation of k-anonymity, several models such as p-sensitive k-anonymity [16], (p^+, α)-sensitive k-anonymity [14], l-diversity [8], (α, k)-anonymity [19] and t-closeness [7] were proposed in the literature in order to deal with the problem of k-anonymity. The work presented in this paper is based on l-diversity model, introduce by [8]. The main contribution of [8] is to introduce the l-diversity property, which provides privacy even when the data publisher does not know what kind of knowledge is possessed by the adversary. Most of the existing solutions are limited only to static data release. That is, in such solutions it is assumed that the entire dataset is available at the time of release. Nevertheless, large microdata sets containing private information are time-evolving, meaning that new data are collected and added, and old data are purged.

One possible method is to publish anonymizations of current microdata, that is, when the new anonymous versions of such a dataset are prepared for release, the current solution is to reprocess the entire dataset, without relying on previous releases of the dataset. However, processing a large dataset in this way to achieve the privacy requirement is time-consuming. Another approach is to anonymize and publish new records periodically. Then researchers can either study each released dataset independently or merge multiple datasets together for more comprehensive analysis. Although straightforward, this approach may suffer from severely low data quality.

The incremental updates are not well addressed in the previous studies. [17] studies the incremental update issue for k-anonymity model. In this paper, we discuss about the updating technique for large time-evolving microdata on l-diversity model which extend the results in [17]. we propose an updating technique for the maintenance of l-diverse large evolving datasets. Essentially, the proposed technique produces a l-diverse dataset starting from a previous l-diverse release solution for the dataset, which is updated to include the new data in the increment dataset and to delete the obsolete dataset. The anonymous process tries to

Table 1. Microdata

MCN	Gender	Age	Zip	Diseases
∗	Male	25	4350	Hypertension
∗	Male	23	4351	Hypertension
∗	Male	22	4352	Depression
∗	Female	28	4353	Chest Pain
∗	Female	34	4352	Obesity
∗	Female	31	4350	Flu

Table 2. 3-anonymous Microdata

MCN	Gender	Age	Zip	Diseases
∗	Male	22-25	435∗	Hypertension
∗	Male	22-25	435∗	Hypertension
∗	Male	22-25	435∗	Depression
∗	Female	28-34	435∗	Chest Pain
∗	Female	28-34	435∗	Obesity
∗	Female	28-34	435∗	Flu

minimize information loss. As our experimental results show, this updating technique is far more efficient than to re-process the whole updated microdata.

2 Preliminaries

Let T be the initial microdata table and T' be the released microdata table. T' consists of a set of tuples over an attribute set. The attributes characterizing microdata are classified into the following three categories.

- *Identifier attributes* that can be used to identify a record such as Name and Medicare card.
- *Quasi-identifier (QI) attributes* that may be known by an intruder, such as Zip code and Age. QI attributes are presented in the released microdata table T' as well as in the initial microdata table T.
- *Sensitive attributes* that are assumed to be unknown to an intruder and need to be protected, such as Disease or ICD9Code. Sensitive attributes are presented both in T and T'.

In what follows we assume that the identifier attributes have been removed and the quasi-identifier and sensitive attributes are usually kept in the released and initial microdata table. Another assumption is that the value for the sensitive attributes are not available from any external source. This assumption guarantees that an intruder can not use the sensitive attributes to increase the chances of disclosure. Unfortunately, an intruder may use record linkage techniques [18] between quasi-identifier attributes and external available information to glean the identity of individuals from the modified microdata. To avoid this possibility of privacy disclosure, one frequently used solution is to modify the initial microdata, more specifically the quasi-identifier attributes values, in order to enforce the k-anonymity property.

Definition 1 (Quasi-identifier). *A quasi-identifier (QI) is a minimal set Q of attributes in microdata table T that can be joined with external information to re-identify individual records (with sufficiently high probability.*

Definition 2 (k-anonymity). *The modified microdata table T' is said to satisfy k-anonymity if and only if each combination of quasi-identifier attributes in T' occurs at least k times.*

A QI-group in the modified microdata T' is the set of all records in the table containing identical values for the QI attributes. There is no consensus in the literature over the term used to denote a QI-group. This term was not defined when k-anonymity was introduced [11,15]. More recent papers use different terminologies such as equivalence class [19,8,7] and QI-cluster [16,14].

For example, let the set {Gender, Age, Zip Code} be the quasi-identifier of Table 1. Table 2 is one 3-anonymous view of Table 1 since there are two QI-groups and the size of each QI-group is at least 3. So k-anonymity can ensure that even though an intruder knows a particular individual is in the k-anonymous microdata table T, s/he can not infer which record in T corresponds to the individual with a probability greater than $1/k$.

The k-anonymity property ensures protection against identity disclosure, i.e. the identification of an entity (person, institution). However, it does not protect the data against attribute disclosure. To deal with this problem in privacy breach, the l-diversity model was introduced in [8].

Definition 3 (l-diversity principle). *A QI-group is said to have l-diversity if there are at least l "well-represented" values for the sensitive attribute. A modified table is said to have l-diversity if every QI-group of the table has l-diversity.*

Machanavajjhala *et al.* [8] gave a number of interpretations of the term "well-represented" in this principle:

1. Distinct l-diversity: The simplest understanding of "well represented" would be to ensure there are at least l distinct values for the sensitive attribute in each QI-group. Distinct l-diversity does not prevent probabilistic inference attacks. A QI-group may have one value appear much more frequently than other values, enabling an adversary to conclude that an entity in the equivalence class is very likely to have that value. This motivated the development of the following two stronger notions of l-diversity.

2. Entropy l-diversity: The entropy of a QI-group G is defined to be:

$$Entropy(G) = -\sum_{s \in S} p(G,s) \log p(G,s)$$

in which S is the set of the sensitive attribute, and $p(G,s)$ is the fraction of records in G that have sensitive value s. A table is said to have entropy l-diversity if for every QI-group G, $Entropy(G) \geq log(l)$. Entropy l- diversity is strong than distinct l-diversity. As pointed out in [8], in order to have entropy l-diversity for each QI-group, the entropy of the entire table must be at least $log(l)$. Sometimes this may too restrictive, as the entropy of the entire table may be low if a few values are very common. This leads to the following less conservative notion of l-diversity.

3. Recursive (c,l)-diversity: Recursive (c,l)-diversity makes sure that the most frequent value does not appear too frequently, and the less frequent values do not appear too rarely. Let m be the number of values in a QI-group, and r_i,

$1 \leq i \leq m$ be the number of times that the i^{th} most frequent sensitive value appears in a QI-group G. Then G is said to have recursive (c, l)-diversity if $r_1 < c(r_l + r_{l+1} + ... + r_m)$. A table is said to have recursive (c, l)-diversity if all of its QI-groups have recursive (c, l)-diversity.

In this paper, we adopt the first interpretation of l-diversity, that is, we say a microdata satisfies l-diversity principle, if there are at least l distinct values in each QI-group. We applied the cluster technique reported in [4]. To ensure that l-diversity is correctly enforced, two constraints are required when the clustering process is performed. First, each resulted cluster must have at least l distinct values for the sensitive attribute. If it does, the subsequent generalization of the cluster elements to a common tuple ensures the l-diversity requirement. Second, the clustering method must act towards minimizing the information loss. The clusters should be formed such that the information lost by generalizing each group of tuples to a common value will be as low as possible.

Definition 4. *[4] [Information Loss] Let $cl \in \mathcal{P}$ be a cluster, $gen(cl)$ its generalization information and $A = \{N_1, \cdots N_s, C_1, \cdots C_t\}$ the set of quasi identifier attributes. The information loss caused by generalizing cl tuples to $gen(cl)$ is:*

$$IL(cl) = |cl|(\sum_{j=1}^{s} \frac{|gen(cl)[N_j]|}{|[min_{r \in T} r[N_j], max_{r \in T} r[N_j]]|}) + \sum_{j=1}^{t} \frac{h(\wedge(gen(cl)[C_j]))}{h(H_{C_j})}$$

where $|cl|$ denotes the cardinality of cluster cl; $|[i_1, i_2]|$ is the size of the interval $[i_1, i_2]$ (the value $i_2 - i_1$); $\wedge(w), w \in H_{C_j}$ is the sub-hierarchy of H_{C_j} rooted at w; $h(H_{C_j})$ denotes the height of the tree hierarchy H_{C_j}.

Definition 5. *Total information loss for a solution $\mathcal{P} = \{cl_1, \cdots, cl_v\}$ of the l-diversity by clustering problem, denoted by $IL(\mathcal{P})$, is the sum of the information loss measure for all the clusters in $\mathcal{P}$.*

The information loss measure penalizes each tuple with a cost proportional with how "far" the tuple is from the cluster generalization information. Intuitively, the smaller the clusters are in a solution and the more similar the tuples in those groups will be, then less information will be lost. So, the desideratum is to group together the most similar objects (i.e. that cause the least possible generalization) in clusters with respect to the l-diversity requir.

3 Dynamic Updating Time-Evolving Microdata

Let $\mathcal{P} = \{cl_1, \cdots cl_v\}$ be a solution for the l-diversity problem for the microdata T. There are three problems arisen in the updating process. The first is that when there is a new segment of data that needs to be added to the original microdata, and how to process the update to make it preserve the l-diversity. The second is when parts of the original data needs to be deleted, how to maintain l-diversity. The third one is a hybrid version of adding and deleting. We can solve the third one by independently solve the first and second problem. The first and second problems are described as follows:

Problem 1. The dataset $\triangle^+T$ is added to T. How to efficiently update $\mathcal{P}$ to $\mathcal{P}' = \{cl'_1, \cdots cl'_v\}$ that ensures l-diversity for $T \cup \triangle^+T$?

Problem 2. The dataset $\triangle^+T$ is deleted from T. How to efficiently update $\mathcal{P}$ to $\mathcal{P}' = \{cl'_1, \cdots cl'_v\}$ that ensures l-diversity for $T - \triangle^-T$?

The solution to the first problem is as follows. Each tuple r in $\triangle^+T$ is added to that cluster in $\mathcal{P}$ that, increased with r, will produce the minimum increase of total information loss. Due to multiple insertions, when a cluster grows bigger than $2k$ elements and it has at least $2l$ distinct sensitive values, we can split that cluster into two sub-cluster in a greedy manner that tries to minimize total information loss.

The solution to the second problem proceeds as follows. Each tuple r in $\triangle^-T$ is deleted from the cluster currently containing it. The clusters that remain with less than k elements or less than l distinct sensitive values are dispersed into the other cluster, in order to maintain l-diversity for $T - \triangle^-T$. Each element r of cl'_j is relocated to another cluster will produce the minimum increase of the total information loss. If a cluster grows bigger then $2k$ elements and with more than $2l$ distinct sensitive values, that cluster will be split into two, which is the same process as the first problem.

Theorem 1. *Let T be a set of records and l be the specified anonymity requirement. Every cluster that the algorithm finds has at least l distinct sensitive values, but no more than $2l - 1$.*

Proof. As the algorithm finds a cluster with the number of sensitive attribute values of the records is equal to or greater than l, every cluster contains at least l distinct sensitive values. If there is one cluster with less than l distinct sensitive values, each record in this cluster could be relocated to other cluster. That is, in the worst case, the records with $l - 1$ distinct sensitive values are added to another single cluster which already has records with l distinct sensitive values. Therefore, the maximum number of distinct sensitive values in a cluster is $2l - 1$.

4 Experimental Results

In our experiment, we adopted the publicly available data set, Adult Database, at the UC Irvine Machine Learning Repository [10], which has become the benchmark of this field and was adopted by [6,8,5]. In this section we compare, in terms of efficiency, scalability, and results quality, the static algorithm from [16] with our incremental algorithms. The algorithms have been implemented and executed on P4 machine with 2.4 GHz each and 1 GB of RAM.

Table 3 provides a brief description of the data including the attributes we used, the type of each attribute data, the number of distinct values for each attribute, and the height of the generalization hierarchy for each attribute. In all the experiments, we considered Age as the set of numerical quasi-identifier attributes, and Work-class, Marital-status, Occupation, Race, Sex, and Native-country as the set of categorical quasi-identifier attributes. l-diversity property

Table 3. Features of Quasi-identifier

Attribute	Type	Distinct values	Height
Age	Numeric	74	5
Workclass	Categorical	8	3
Education	Categorical	16	4
Country	Categorical	41	3
Marital Status	Categorical	7	3
Race	Categorical	5	3
Gender	Categorical	2	2

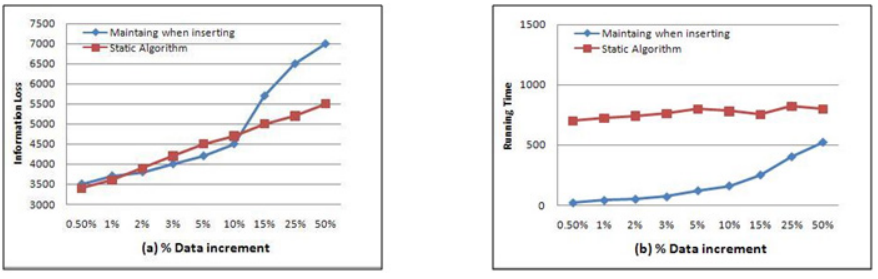


Fig. 1. Information Loss vs Running Time (I)

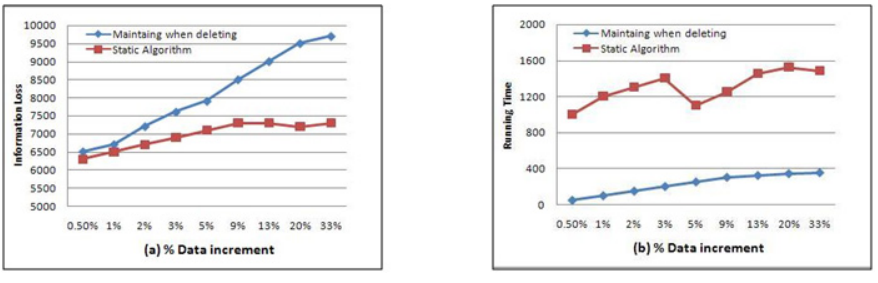


Fig. 2. Information Loss vs Running Time (II)

was enforced in respect to the quasi-identifier consisting of all these seven attributes. We removed all tuples that contained the unknown value for one or more of the quasi-identifier attributes from the data.

The experiment contains three steps. First, the static algorithm from [16] was applied on a dataset, which is a subset extracted from the entire adult dataset. Second, we applied the dynamic algorithm to update the clusters produced by the static algorithm and considering several different choices of inserting/deleting dataset. Third, the static algorithm was applied on the entire new updated dataset datasets. When doing inserting, T has 10000 objects, and the inserting dataset had different sizes, varying between 0.5% and 50% of the entire adult

dataset. When doing deleting, the deleting parts had different sizes, varying between 50 and 5000 tuples. The values considered for l were 2, 4.

In Fig. 1(a) and 2(a), we compare: a) the information loss for each set of clusters obtained by applying the static l-diversity algorithm, followed by updating algorithms on the corresponding updated dataset; b) the information loss for the set of clusters obtained by applying the static algorithm on the updated dataset. We can expect that the information loss obtained by the updating algorithm deteriorates when the increment/ decrement dataset grows in size w.r.t. the initial dataset size. Nevertheless, as is usually the case in the real world databases evolution, for small modification amounts, the information loss remains at about the same level as if we would use the static algorithm. From these experiments, we draw the conclusion that the updating algorithm can be used for maintaining l-diverse microdata when the changing portions of the datasets are small.

Fig. 1(b) and Fig. 2(b) illustrate the running time for the updating algorithms compared with the static algorithm. The time for incrementally processing the datasets grows with the size of the datasets, however, it is still significantly lower than the time required to process the datasets with the static algorithm. Whether to use updating algorithm or a static one is to be decided by the requirement of data quality and execution time. The advantages of dynamic updating algorithms can maintain acceptable data quality while the running time is tolerated.

5 Conclusion and Future Work

In this paper, we identified and investigate the problem of maintaining l-diversity in time evolving microdata, and proposed a simple yet effective solution. Maintaining l-diversity against various types of dynamic updates is an important and practical problem. As we show in experiments, the running time of the dynamic updating algorithms is significantly lower than that of the static algorithm. From the data quality perspective, the information loss is also comparable with the information loss obtained by applying the non-incremental algorithm to the final dataset. As future work, we will make more comprehensive experimental studies to compare the dynamic method with others and extend to other privacy paradigms.

Acknowledgement

Thanks for Professor Traian Marius Truta for his valued suggestion. This research was funded by Australian Research Council (ARC) grant DP0774450 and DP0663414.

References

1. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing tables. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 246–258. Springer, Heidelberg (2005)
2. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Approximation algorithms for k-anonymity. Journal of Privacy Technology, paper number 20051120001

3. Bayardo, R., Agrawal, R.: Data privacy through optimal k-anonymity. In: Proceedings of the 21st International Conference on Data Engineering, ICDE (2005)
4. Byun, J., Kamra, A., Bertino, E., Li, N.: Efficient k-Anonymization using Clustering Techniques. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 188–200. Springer, Heidelberg (2007)
5. Fung, B., Wang, K., Yu, P.: Top-down specialization for information and privacy preservation. In: Proc. of the 21st International Conference on Data Engineering (ICDE 2005), Tokyo, Japan (2005)
6. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: Efficient Full-Domain k-Anonymity. In: ACM SIGMOD International Conference on Management of Data (June 2005)
7. Li, N., Li, T., Venkatasubramanian, S.: t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In: ICDE 2007, pp. 106–115 (2007)
8. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-Diversity: Privacy beyond k-anonymity. In: ICDE 2006 (2006)
9. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: Proc. of the 23rd ACM-SIGMOD-SIGACT-SIGART Symposium on the Principles of Database Systems, Paris, France, pp. 223–228 (2004)
10. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases, University of Califonia, Irvine (1998), `www.ics.uci.edu/~mlearn/MLRepository.html`
11. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
12. Sun, X., Li, M., Wang, H., Plank, A.: An efficient hash-based algorithm for minimal k-anonymity problem. In: 31st Australasian Computer Science Conference (ACSC 2008), Wollongong, NSW, Australia, pp. 101–107. CRPIT 74 (2008)
13. Sun, X., Wang, H., Li, J.: On the complexity of restricted k-anonymity problem. In: Zhang, Y., Yu, G., Bertino, E., Xu, G. (eds.) APWeb 2008. LNCS, vol. 4976, pp. 287–296. Springer, Heidelberg (2008)
14. Sun, X., Wang, H., Li, J., Traian, T.M., Li, P.: (p^+, α)-sensitive k-anonymity: a new enhanced privacy protection model. In: 8th IEEE International Conference on Computer and Information Technology (IEEE-CIT 2008), July 8-11, pp. 59–64 (2008)
15. Sweeney, L.: k-anonymity: A Model for Protecting Privacy. International Journal on Uncertainty Fuzziness Knowledge-based Systems 10(5), 557–570 (2002)
16. Traian, T.M., Bindu, V.: Privacy Protection: l-diversity Property. In: International Workshop of Privacy Data Management (PDM 2006); In Conjunction with 22th International Conference of Data Engineering (ICDE), Atlanta (2006)
17. Truta, T.M.: Alina Campan, k-Anonymization Incremental Maintenance and Optimization Techniques. In: ACM Symposium on Applied Computing (SAC 2007), special track on Data Mining, Seoul, Korea (2007)
18. Winkler, W.E.: Advanced Methods for Record Linkage. In: Proceedings of the Section on Survey Research Methods, American Statistical Society, pp. 467–472 (1994)
19. Wong, R., Li, J., Fu, A., Wang, K.: (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: KDD 2006, pp. 754–759 (2006)

Knowledge Discovery from Honeypot Data for Monitoring Malicious Attacks

Huidong Jin[1,2], Olivier de Vel[3], Ke Zhang[1,2], and Nianjun Liu[1,2]

[1] NICTA Canberra Lab, Locked Bag 8001, Canberra ACT, 2601, Australia
huidong.jin@nicta.com.au,ke.zhang@cecs.anu.edu.au,
nianjun.liu@nicta.com.au
[2] RSISE, the Australian National University, Canberra ACT, 0200, Australia
[3] Command, Control, Communications and Intelligence Division, DSTO, PO Box 1500, Edinburgh SA 5111, Australia

Abstract. Owing to the spread of worms and botnets, cyber attacks have significantly increased in volume, coordination and sophistication. Cheap rentable botnet services, e.g., have resulted in sophisticated botnets becoming an effective and popular tool for committing online crime these days. Honeypots, as information system traps, are monitoring or deflecting malicious attacks on the Internet. To understand the attack patterns generated by botnets by virtue of the analysis of the data collected by honeypots, we propose an approach that integrates a clustering structure visualisation technique with outlier detection techniques. These techniques complement each other and provide end users both a big-picture view and actionable knowledge of high-dimensional data. We introduce KNOF (K-nearest Neighbours Outlier Factor) as the outlier definition technique to reach a trade-off between global and local outlier definitions, i.e., K^{th}-Nearest Neighbour (KNN) and Local Outlier Factor (LOF) respectively. We propose an algorithm to discover the most significant KNOF outliers. We implement these techniques in our hpdAnalyzer tool. The tool is successfully used to comprehend honeypot data. A series of experiments show that our proposed KNOF technique substantially outperforms LOF and, to a lesser degree, KNN for real-world honeypot data.

Keywords: Knowledge discovery, outlier detection, density-based cluster visualisation, botnet, honeypot data, Internet security.

1 Introduction

The Internet has evolved into a platform for a large number of security-sensitive services and applications. Online banking and payment, e.g., are now ubiquitous [1]. Cyber attacks have significantly increased in volume, coordination and sophistication with the increase in online attacks from a variety of attack vectors such as worms and botnets. A botnet is a set of compromised computers subject to a common Command-and-Control mechanism, often used for nefarious purposes. Each compromised computer, i.e., a bot, behaves like a complex

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 470–481, 2008.

of worms, rootkits and Trojan horses [1]. The bot can automatically exploit, propagate and initiate a remote control channel to its master. A flourishing industry of botmasters sell or rent stealthy botnet services, which is compromising Internet security.

A 'honeypot' is a computer or network trap dedicated to attracting and monitoring malicious computer attacks, more recently botnet attacks. Various honeypots have successfully been used in cyber attack surveillance, analysis or defense [1,2,3]. They have collected large amount of Internet traffic [4] and security experts have made great efforts to comprehend honeypot data manually [5,1] or via simple tools [4,6].

Automatic knowledge discovery from honeypot data is still in its infancy, especially for proactive security purposes such as detecting zero-day attacks or discovering new bots as they are deployed. In this paper, we report our efforts on developing techniques and tools to analyse honeypot data for proactive security. By integrating clustering structure visualisation together with outlier detection, our hpdAnalyzer tool can not only provide a big picture of honeypot data patterns, but also identify some abnormal activities which security experts can choose for further action. We also combine a global outlier definition, K-Nearest Neighbour (KNN), with a local outlier definition, Local Outlier Factor (LOF), which we call KNOF, as the outlier definition in honeypot data. The tool has been tested on real-world honeypot data.

The remaining paper is organised as follows. We outline the relevant data mining techniques in Section 2. We describe the KNOF algorithm and the hpdAnalyzer tool in Sections 3 and 4 respectively. Experimental results are presented in Section 5 and these are followed by concluding comments in Section 6.

2 Related Data Mining Techniques

To analyse honeypot data, we propose to use two kinds of data mining techniques. First, we conduct an exploratory data analysis to obtain an overall understanding of the data set. This is useful for gaining a high-level understanding of the way the data are structured, e.g., inherent clusters. Furthermore, we detect outliers to highlight abnormalities within the data, like suspicious behaviours or attacks. Such outliers can then be selected to undertake further analysis. Visualising outliers properly in the first phase can help us verify/choose suitable outlier detection techniques. Moreover, outliers may help us clarify clustering structure easily.

We first outline a density-based cluster visualisation technique, OPTICS (Ordering Points To Identify the Clustering Structure) [7]. Density-based clusters are regarded as regions where objects are dense and which are separated by sparse regions (which may contain outliers). They may have arbitrary shape. OPTICS does not produce a clustering of a data set D explicitly, but instead creates an augmented *ordering* of the data set representing its density-based clustering structure. The cluster-ordering could be used to visualise the inherent clustering structure of a high-dimensional data set.

OPTICS has two parameters, ϵ and k. The generating distance ϵ indicates the maximum radius of the neighbourhood. OPTICS ignores neighbours that are located further away than ϵ. The ϵ-neighbours of an object p is defined as $\mathcal{S}_\epsilon(p) = \{q \in D \,|dis(p,q) \leq \epsilon\}$ where $dis(p,q)$ is the distance between objects p and q. The second parameter k is a natural number and indicates the minimal number of neighbours within its ϵ-neighbours for an object to become a **core object** in clustering. That is, p is a core object if and only if $|\mathcal{S}_\epsilon(p)| \geq k$. For a given data set, its core objects constitute the core part of clusters. The **core-distance** is the smallest distance $\varepsilon(\leq \epsilon)$ between p and an object in its ϵ-neighbourhood such that p would be a core object, i.e., $\left|\mathcal{S}_{\text{core-dis}(p)}\right| \geq k$. The core-distance of a non-core object q is UNDEFINED. Then, the **reachable-distance** of p w.r.t. another object o is formalised as

$$\mathbf{r}_{\epsilon,k}(p,o) = \begin{cases} \text{UNDEFINED}, & \text{if } |\mathcal{S}_\epsilon(o)| < k; \\ \max\{\text{core-dis}(o), dis(o,p)\}, & \text{otherwise.} \end{cases} \quad (1)$$

OPTICS [7] creates a clustering-ordering, additionally storing the core-distance and a suitable reachable-distance for each object as follows. It starts with any object p. It retrieves the ϵ-neighbours, i.e., $\mathcal{S}_\epsilon(p)$. If p is a core object, the core distance of p is calculated w.r.t. k. Object p is assigned an order which starts from a value equal to one. The object's reachable-distance is also stored. All the ϵ-neighbours are put into a priority queue with undefined reachable-distance. OPTICS then calculates and updates the reachable-distance of objects in the priority queue w.r.t. this core object p. The objects in the queue will be sorted in the ascending order of their current reachable-distance. OPTICS will select the object at the head of the queue to process. If p is not a core object, p is assigned an order and its reachable-distance is stored. However, neither its ϵ-neighbours are processed, nor the reachable-distance of objects within the queue is updated. The consecutive object in the queue is then chosen as the next object. When the priority queue is empty and the data set has unprocessed objects, a new random unprocessed object will be selected. Thus, the OPTICS algorithm generates an ordering of the objects $\mathbf{o} : \{1..n\} \rightarrow D$ and corresponding reachable-distances $\mathbf{r} : \{1..n\} \rightarrow R_{\geq 0}$. This information is sufficient to extract all density-based clusterings w.r.t. any distance $\varepsilon \leq \epsilon$. For clustering structure visualisation, we employ the reachability plot where the reachable-distance values $\mathbf{r}$ are plotted for each object in the cluster-ordering $\mathbf{o}$. According to OPTICS, clusters correspond to valleys in the reachability plot as exemplified in the top window in Fig. 1(a)

There are various definitions of outliers and associated outlier detection techniques, including statistical distribution-based, depth-based, clustering-based, distance-based and density-based techniques [8,9]. For high-dimensional data, we use the distance-based K^{th}-Nearest Neighbour (KNN) [10] technique and the density-based outlier detection technique, LOF [11].

A KNN outlier [10] does not require any apriori knowledge of data distributions. It relies on the distance of the k^{th} nearest neighbour of an object p, denoted as $\mathcal{D}^k(p)$, also known as the k-distance [11]. Intuitively, objects with

larger values of $\mathcal{D}^k(p)$ have more sparse neighbourhoods and are thus typically stronger outliers than objects belonging to dense clusters which will tend to have lower values of $\mathcal{D}^k(p)$. Given two parameters k and n, an object p is an outlier if no more than $n-1$ other objects in the data set have a higher value for $\mathcal{D}^k$ than p. In other words, the top n objects with the maximal $\mathcal{D}^k$ are regarded as outliers. We simply refer these top-n KNN outliers, hereafter as "KNN outliers", when this is clear from the context. Because the definition takes a global view of a data set, these outliers can be viewed as 'global outliers' [11].

Density-based outliers are objects that are outlying relative to their local neighbourhoods, particularly w.r.t. the densities of the neighbourhoods. These outliers are regarded as 'local' outliers, such as defined by the Local Outlier Factor (LOF) [11]. LOF needs a parameter k. To simplify our discussion, we assume that less than k objects are identical in a data set. The k-distance neighbourhood of an object p, $\mathcal{N}_k(p)$, comprises the objects whose distance from p is not greater than $\mathcal{D}^k(p)$. The *reachability distance* of p w.r.t. another object o, $reach\_dist_k(p,o)$, is the maximum of $\mathcal{D}^k(o)$ and the distance between p and o, $dis(p,o)$. The **local reachability density** of an object p is the inverse of the average reachability distance based on the k nearest neighbours of p, i.e.,

$$lrd_k(p) = \frac{1}{\frac{\Sigma_{o \in \mathcal{N}_k(p)} reach\_dist_k(p,o)}{|\mathcal{N}_k(p)|}}. \tag{2}$$

The **Local Outlier Factor** (LOF) of an object p is the ratio of the average local reachability density of p's k-nearest neighbours to the local reachability desnity of p. Intuitively, the lower an object p's local reachability density is, and the higher those of p's k-nearest neighbours are, the higher is p's LOF value. Formally,

$$LOF_k(p) = \frac{\Sigma_{o \in \mathcal{N}_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|\mathcal{N}_k(p)|}. \tag{3}$$

3 K-Nearest Neighbours Outlier Factor

In this paper we will concentrate on using the top n outliers as an outlier definition. There are several considerations. (1) Firstly, it is easy to set the unique parameter n. (2) Secondly, it is usually computationally cheap. (3) Finally, it greatly facilitates our experiments as there are no labeling information for outliers in most, especially unseen, real-world data sets.

As discussed above, distance-based outlier definitions like KNN prefer 'global' outliers which lie away from other objects. LOF prefers 'local' outliers whose neighbours have much higher density. Our preliminary experiments substantiate that both LOF and KNN perform reasonably well. However, the intersection of the top n KNN and the top n LOF outliers is only a small proportion of n. Only a few outliers highlighted by LOF (or KNN) are of great interest for the honeypot data from proactive security viewpoint. Some examples will be

given in Section 5. To make better use of these two "outlierness" definitions, especially for highlighting the top n outliers of practical interest, we propose a new "outlierness" measure. It combines these two measures in order to reach a good balance between 'global' and 'local' outlierness. A large number of the top n outliers under this definition is found in the top n KNN outliers. It also corresponds to a number of the top n LOF outliers. This definition could be used for specific applications as shown in Section 5. We refer to the new measure as the k-nearest Neighbours Outlier Factor (KNOF).

Definition 1. *For a given integer k, the* **KNOF** *of an object p is the product of its LOF value and the average distance from p to each of the objects in the k-nearest neighbourhood.*

$$KNOF_k(p) = LOF_k(p) \times \overline{d_{\mathcal{N}_k}}(p) \triangleq LOF_k(p) \times \frac{\Sigma_{o \in \mathcal{N}_k(p)} dis(p,o)}{|\mathcal{N}_k(p)|}. \quad (4)$$

There are several reasons for us to define KNOF as the product of the LOF and the average distance $\overline{d_{\mathcal{N}_k}}(p)$ (roughly KNN) values. Firstly, the KNN and the LOF values have quite different value ranges. As illustrated in Fig. 1, KNN could be as large as 20. The maximum of KNN could be three or more times larger than that of LOF. A simple sum or average of KNN and LOF would be predominated by KNN, especially for top n outliers. Such measures can't reach a good balance between KNN and LOF. Secondly, as we concentrate on the top n outliers, this definition of KNOF has the same performance as a lot of other measures such as normalisation(KNN)*normalisation(LOF), sqrt(KNN)*sqrt(LOF) or ln(KNN)+ln(LOF), but KNOF defined by Eq.(4) is relatively simple. Thirdly, based on our preliminary experiments, other measures like sqrt(KNN)*LOF or KNN*sqrt(LOF) could not empirically perform as well as the KNOF defined above.

The algorithm that obtains the top n KNOF$_k$ outliers for all the N objects in a given data set D is outlined in Algorithm 1. In Step 1, a key component is to retrieve the κ-nearest neighbours for every object p. To reduce the number of retrieval operations, we only store a set of objects q such that $dis(q,p) \leqslant \mathcal{D}^k(p)$ as its k-distance neighbourhood. That is, we restrict the cardinality of $\mathcal{N}_k(p)$ to κ ($\geq k$) which is $2k$ by default. To retrieve the κ-nearest neighbours, without any index support, a sequential scan through the whole data set has to be performed. In this worst case, the run-time of this step is $O(N^2)$. If a tree-based spatial index such as R^*-tree is used [7,11], the run-time is reduced to $O(N \log N)$.

Steps 2 and 3 are straightforward and calculate LOF and KNOF values according to Eqs. 3 and 4 respectively. As the k-distance neighbourhood is materialised, these steps are linear with N. Step 4 sorts the N objects according to their KNOF values. Its computation complexity could be $O(N \log N)$. Thus, the overall computation complexity of Algorithm 1 is $O(N \log N)$ with appropriate index support. The worst case is $O(N^2)$.

Algorithm 1. Top-n KNOF(Top-n K-nearest Neighbours Outlier Factor)

Input: A given data set D, natural numbers n, k and $\kappa(\geq k)$. κ is by default $2k$.

1. For each object p in D, retrieve p's κ-nearest neighbours, compute p's k-distance $\mathcal{D}^k(p)$, store the objects whose distance from p is not greater than $\mathcal{D}^k(p)$ as its k-distance neighbourhood $\mathcal{N}_k(p)$, and compute $\overline{d_{\mathcal{N}_k}}(p)$;
2. Calculate the local reachability density for each object p and then calculate the LOF value according to Eq. 3;
3. Compute the KNOF value for each object p, $\text{KNOF}(p) = \text{LOF}(p) \times \overline{d_{\mathcal{N}_k}}(p)$.
4. Sort the objects according to their KNOF values;
5. **Output:** the first n objects with the highest KNOF values.

4 The Honeypot Data Analyser: hpdAnalyzer

We have implemented the four techniques described above, namely, OPTICS, LOF, KNN and KNOF in our honeypot data analyser, called hpdAnalyzer.

Besides normal data sets, the hpdAnalyzer tool can also handle network files. We use the winPcap or libPcap packages to capture and transmit network packets. To facilitate feature generation, a connection is considered 'closed' if (1) either side of the connection has sent an RST packet, or (2) either side has sent the last ACK packet while the other side has moved to the TIME_WAIT state. A connection is 'active' if it is not closed. A 'new' connection is established if none of existing 'active' connections has the same socket. The hpdAnalyzer tool generates three categories of features for each connection [12] namely, (a) 15 features are **content-based** and extracted directly from tcpdump. For example, num_SYNs_src represents the number of SYN packets flowing from source to destination and num_bytes_src is the number of data bytes flowing from source to destination. (b) Four **time-based** features attempt to capture previous connections with similar characteristics [13]. They only consider the connections in the past user-defined t window, such as the number of connections made by the same source as the current connection in the last t seconds. The parameter t is user-defined. (c) Four **connection-based** features capture similar characteristics of the last c connections from the same source, such as the number of connections made by the same source as the current record in the last c connections. The parameter c is user-defined.

In real-world data sets, some features may have much larger values than other features. Thus, without scaling, the outliers would probably be determined by those features. To avoid this, hpdAnalyzer normalises all of the features to one range $[0, 1]$. We implement two different methods for normalisation: linear and nonlinear. Nonlinear normalisation is a piecewise linear mapping with one turning point. The turning point is mapped onto 0.5. We choose the turning point such that, when greater than this particular value, increasing the value does not make any significant change with respect to outlierness. For example, by examining the sample data, we find that any connection having the metric num_bytes_src greater than 100,000 is considered fairly suspicious for honeypot data. There is little difference

in term of suspiciousness for a connection with num_bytes_src equal to 100,000 or 200,000. Thus, the turning-point for this feature would be 100,000.

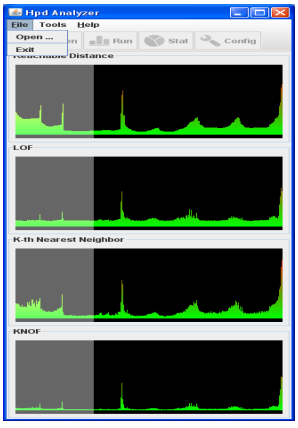
End users can interact with the hpdAnalyzer tool via both a Command Line Interface (CLI) and a Graphic User Interface (GUI). The CLI provides several useful commands to work with data such as setting parameters, loading data, normalising data, listing the top n outliers, and saving intermediate results. It provides an optional *data window* for 2-D visualisation, e.g., Fig. 1(b) illustrates the DB4 data set from [7]. The GUI provides data visualisation and interaction capabilities. For example, the *main window*, as exemplified in Fig. 1(a), shows all values of the reachable-distance, LOF, KNN and KNOF outlierness in the cluster-ordering generated by OPTICS. Some details of these plots can be shown in one or more *display windows*. For example, Fig. 1(c)-1(f) illustrate the shadowed parts in Fig. 1(a). In these *display windows*, we can interactively choose a specific object. If we choose the object as indicated by the long vertical line in Fig. 1(c), the other three *display windows* will also indicate this object simultaneously, as illustrated in Fig. 1(d)-1(f). At the same time, the details for that object are listed in the CLI. In the optional *data window*, that object is highlighted by expanding white circles, as shown in Fig. 1(b). It is easy to see that the reachability plot illustrates the intrinsic cluster structure of DB4, and the three outlierness values show the details within clusters and between two clusters. In the first cluster indicated by the first valley in Fig 1(c), objects (bottom-left in Fig 1(b)) are scattered with similar local density as observable in Fig. 1(d), but some have higher KNN values as shown in Fig. 1(e). As illustrated in Fig. 1(f), the KNOF plot is smoother than the KNN plot for this cluster.

5 Experimental Results on Real-World Honeypot Data

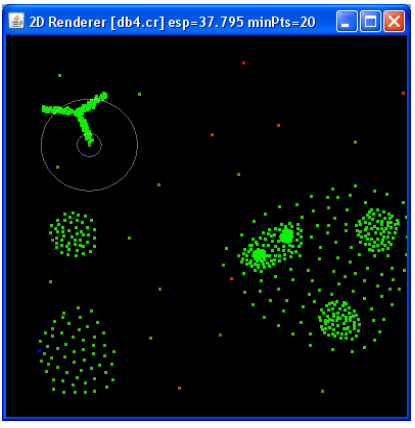
We have tested our hpdAnalyzer tool, especially the three outlier detection techniques, on several data sets collected by honeypots. We report the results on the SotM 28 data collected by the Mexico Honeynet project on the honeypot with IP 192.168.100.28. The data contain two binary network files and records a successful IRC-based bot attack. The Day1 data (day1.log) has 18,843 packets and records the procedure of a break-in. The Day3 data (Day3.log) has 123,123 packets and captures unique activities following the compromise.

For the honeypot data, we don't have labeling information about the suspiciousness or outlierness of any network connection. Thus, we could not use detection accuracy or precision as evaluation metrics. We simply examine the top n outliers to compare the performance of the three outlier detection techniques. Once they generate results, we ask a network expert to manually assign suspiciousness values to these network connections in the context of IRC-based bot attacks [1], without disclosing the algorithms. The suspiciousness value ranges from 0 to 1. The sum of the suspiciousness of the top n connections serves as an indicator for outlier detection performance for these real-world data.

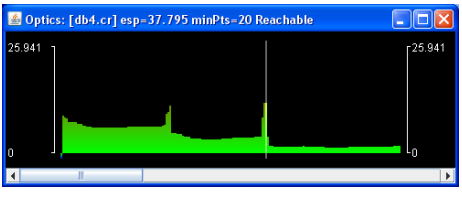
We first discuss some results of hpdAnalyzer for the Day1 data. Fig. 2 illustrates typical results for Day1 from hdpAnalyzer with $k = 18$, $c = 10$ and $t = 500$



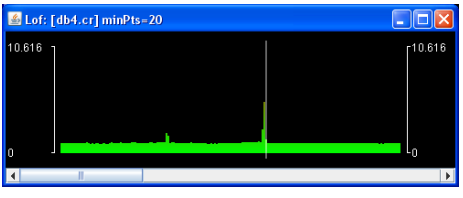
(a) Main window for DB4



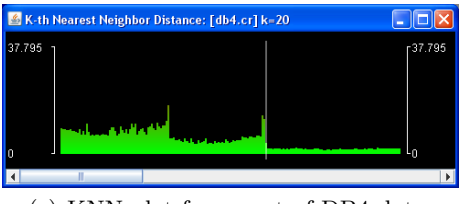
(b) 2-D visualisation of original DB4.



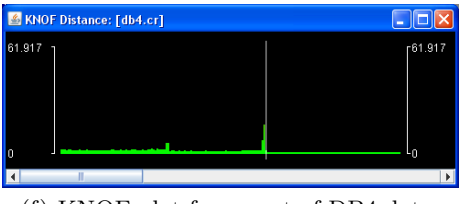
(c) Reachability plot for partial DB4 data.



(d) LOF plot for a part of DB4 data.



(e) KNN plot for a part of DB4 data.

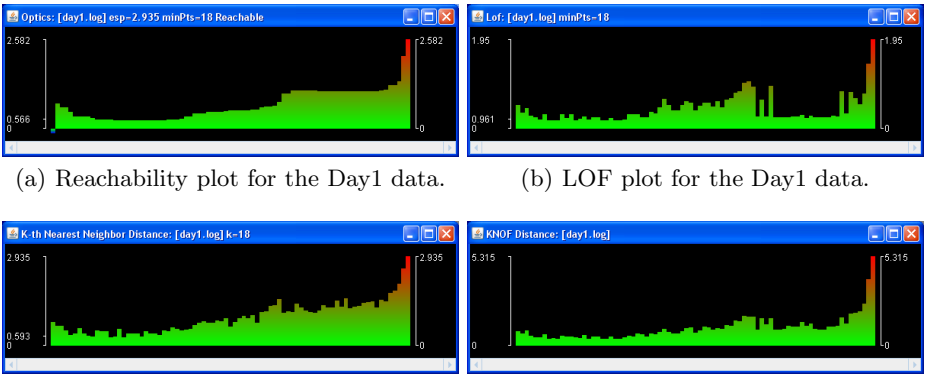


(f) KNOF plot for a part of DB4 data.

Fig. 1. Results of hpdAnalyzer on a synthetic 2-D DB4 data set from [7]

seconds. In these four subfigures, the Y-axis indicates the value of reachable-distance or outlierness. The ordering of network connections along the X-axis is generated by OPTICS. Combining Figs. 2(a) and 2(b), we can observe two clusters, one with high local density and the other with relatively low local density. All of the three outlier detection techniques indicate that there are some outliers at the end of cluster-ordering. However, LOF (Fig. 2(b)) indicates most outliers are located between the two clusters, KNN (Fig.2(c)) indicates most outliers in the second cluster, while KNOF (Fig. 2(d)) indicates there are some outliers in the second cluster and some between the two clusters.

Table 1 lists the top 10 connections highlighted by KNOF for the Day1 data. The three outlier techniques have the same top 2 suspicious connections. KNOF only shares the first five connections with KNN, and seven connections with LOF. Clearly, KNOF reaches a trade-off between the LOF and KNOF techniques.



(a) Reachability plot for the Day1 data.

(b) LOF plot for the Day1 data.

(c) KNN plot for the Day1 data.

(d) KNOF plot for the Day1 data.

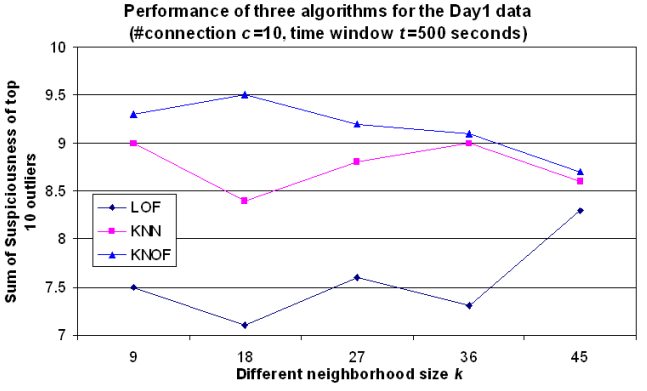
Fig. 2. Experimental results for the Day1 data ($t = 500s$, $c = 10$ and $k = 18$)

Table 1. The top 10 connections highlighted by KNOF for the Day1 data

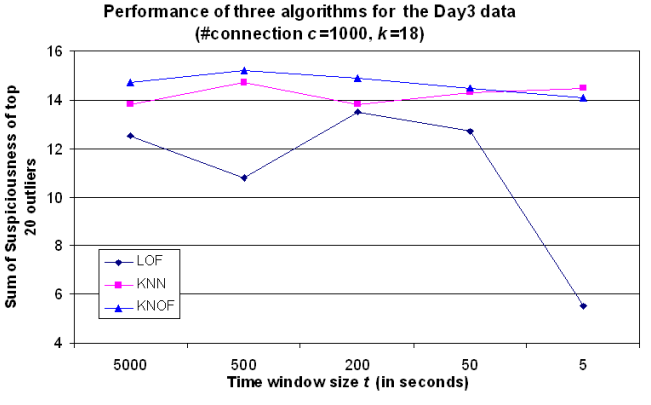
Rank			Outlierness			First Last packets	sockets	suspiciousness
KNOF	LOF	KNN	KNOF	LOF	KNN			
1	1	1	5.31	1.95	2.93	8621 16890	192.168.100.28:32802 – 206.252.192.195:5555	1.0
2	2	2	3.97	1.66	2.58	588 8351	61.219.90.180:56712 – 192.168.100.28:1524 (ingreslock)	1.0
3	>10	3	2.48	1.29	2.13	1038 5318	192.168.100.28:32789 – 62.211.66.53:80 (www)	1.0
4	>10	4	2.03	1.16	1.95	5601 8280	192.18.99.122:20(fpt-data) – 192.168.100.28:32794	1.0
5	>10	5	1.97	1.23	1.88	996 1008	62.211.66.16:20 (ftp-data) – 192.168.100.28:32788	1.0
6	9	>10	1.89	1.30	1.71	8706 9981	80.117.14.44:3934 – 192.168.100.28:7000 (afs3-fileserver)	1.0
7	3	>10	1.73	1.44	1.70	561 8349	61.219.90.180:56399 – 192.168.100.28:6112 (dtspc)	0.5
8	6	>10	1.71	1.37	1.69	12929 12958	80.117.14.44:1047 – 192.168.100.28:7000 (afs3-fileserver)	1.0
9	8	>10	1.68	1.32	1.65	16758 16900	80.117.14.44:2398 – 192.168.100.28:7000 (afs3-fileserver)	1.0
10	5	>10	1.65	1.38	1.63	8362 8708	80.117.14.44:3934 – 192.168.100.28:7000 (afs3-fileserver)	1.0
55	>64	>65	24.45	14.11	19.86	sum		9.5

With the exception of connection number 7, the suspiciousness value for the other nine connections are 1. E.g., Number 1 connection is an IRC conversation that can be used by a botnet for Command & Control. Number 3 connection is for downloading a rootkit (sol.tar.gz) via the http protocol from 62.211.66.53. For this relatively small data set, the sum of suspiciousness of the top 10 connections for KNOF, KNN and LOF are 9.5, 8.4 and 7.1 respectively. KNOF performs somewhat better than the other two.

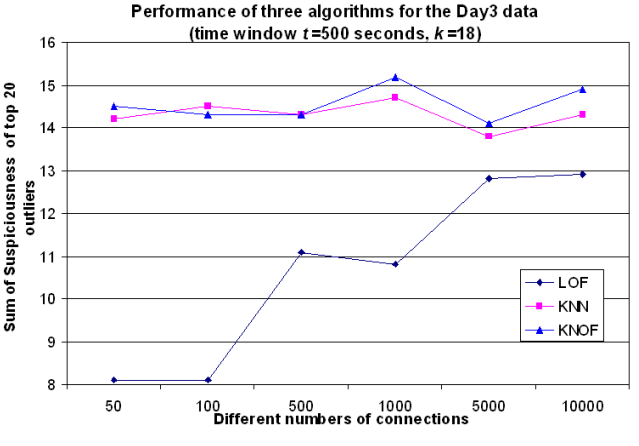
We now describe three sets of comparison results for honeypot data to examine the influence of different neighbourhood size k, time window size t and connection numbers c on the three techniques. The performance of the techniques for the Day1 data is summarised in Fig. 3(a). The Y axis indicates the sum of suspiciousness of the top 10 outliers highlighted by each outlier detection technique. For all the five different k values, KNOF outperforms KNN and LOF, though the difference becomes small when k is very large. On average, the sum of suspiciousness of the top 10 outliers generated by KNOF, KNN and LOF is 9.16, 8.76 and 7.56 respectively. One-tailed paired T-tests indicates there are statistically significant differences among the three techniques at the 0.05 level.



(a) For different neighbourhood size k



(b) For different time window size t



(c) For different connection numbers c

Fig. 3. Performance comparison among the three outlier detection techniques

Fig. 3(b) illustrates the sum of suspiciousness of the top 20 outliers highlighted for the Day3 data with five different time window size t. For all the five different time window sizes, KNOF and KNN perform consistently well. KNOF and KNN clearly outperform LOF. Except for the case $t = 5$, where KNN is favourable, KNOF has the best performance among these three techniques. Averaged over the five settings, the sum of suspiciousness of the top 20 outliers highlighted by KNOF, KNN and LOF is 14.68, 14.22 and 11.00 respectively. Fig. 3(c) shows the performance comparison using difference connection numbers c during the feature generation. Again we consider the sum of suspiciousness of the top 20 outliers. For all the six different settings, KNOF and KNN perform consistently well and outperform LOF. For four of the six settings, KNOF is more favourable. Averaged over the six settings, the sum of suspiciousness of the top 20 outliers highlighted by KNOF, KNN and LOF is 14.55, 14.30 and 10.63, respectively. One-tailed paired t-tests indicate KNOF performs significantly better than LOF at the 0.05 level while better than KNN at the 0.10 level.

6 Conclusion and Discussion

In this work, we have integrated the cluster structure visualisation technique OPTICS with, global and local, outlier detection techniques (KNN and LOF) in our honeypot data analyser tool, hpdAnalyzer. We have proposed and implemented a new outlier detection technique namely, KNOF (K-nearest Neighbour Outlier Factor) which attempts to reach a trade-off between the global and the local outlierness. Experimental results have indicated that the visualisation can complement the outlier detection techniques to highlight outliers and understand the relation between outliers and inherent clusters in data sets. A series of experiments on real-world honeypot data have illustrated that the new proposed outlier detection technique, KNOF, outperforms KNN slightly and LOF substantially.

According to previous work, say [12], LOF performs better on normal network traffic data, especially in comparison with KNN. However, KNN clearly outperforms LOF for honeypot data. Hopefully, experiments on more real-world data sets will provide a better understanding of this. Besides the connection level features, we are looking to use application-level features, e.g., texts/commands generated by software bots that have different characteristics from those used by humans.

Acknowledgements

The authors appreciate the great help and discussion from colleagues and researchers, especially Andrew Clark and George Mohay on recent security progress, Marc Dacier and Corrado Leita on the Leurre.com/SGNet honeypot projects, Phan Thien Son and Lan Du on network security, implementation and experiment issues, Dr. Joerg Sander on OPTICS source codes, Darren Williams for honeypot setting up, and so on. The authors thank the four anonymous reviewers for their

constructive comments and suggestions. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Barford, P., Yegneswaran, V.: An inside look at botnets. In: Book Series Advances in Information Security: Malware Detection, Part III, vol. 27, pp. 171–191 (2007)
2. The Honeynet Project (ed.): Know Your Enemy: Learning about Security Threats, 2nd edn. Addison Wesley Professional, Reading (May 2004)
3. Provos, N., Holz, T.: Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley Professional, Reading (2007)
4. Pouget, F., Dacier, M., Zimmerman, J., Clark, A., Mohay, G.: Internet attack knowledge discovery via clusters and cliques of attack traces. Journal of Information Assurance and Security 1, 21–32 (2006)
5. Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B., Dagon, D.: Peer-to-peer botnets: Overview and case study. In: HotBots 2007 (April 2007) Paper No. 1
6. Nazario, J.: Botnet tracking: Tools, techniques, and lessons learned (2007) (accessed November 14, 2007), `http://www.blackhat.com/presentations/bh-dc-07/Nazario/Paper/bh-dc-07-Nazario-WP.pdf`
7. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: ordering points to identify the clustering structure. In: SIGMOD 1999, pp. 49–60 (1999)
8. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann Publishers, San Francisco (March 2006)
9. Jin, H., Wong, M.L., Leung, K.S.: Scalable model-based clustering for large databases based on data summarization. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(11), 1710–1719 (2005)
10. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: SIGMOD 2000, pp. 427–438 (2000)
11. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: SIGMOD 2000, pp. 93–104 (2000)
12. Lazarevic, A., Ertöz, L., Kumar, V., Ozgur, A., Srivastava, J.: A comparative study of anomaly detection schemes in network intrusion detection. In: SDM 2003 (2003)
13. Lee, W., Stolfo, S.J.: Data mining approaches for intrusion detection. In: USENIX SSYM 1998, p. 6 (1998)

Detecting the Knowledge Boundary with Prudence Analysis

Richard Dazeley[1] and Byeong-Ho Kang[2]

[1] School of Information Technology and Mathematical Sciences,
University of Ballarat, Ballarat, Victoria 7353, Australia
[2] School of Computing and Information Systems,
University of Tasmania, Hobart, Tasmania, 7001
r.dazeley@ballarat.edu.au, bhkang@utas.edu.au

Abstract. Prudence analysis (PA) is a relatively new, practical and highly innovative approach to solving the problem of brittleness in knowledge based systems (KBS). PA is essentially an online validation approach, where as each situation or case is presented to the KBS for inferencing the result is simultaneously validated. This paper introduces a new approach to PA that analyses the structure of knowledge rather than the comparing cases with archived situations. This new approach is positively compared against earlier systems for PA, strongly indicating the viability of the approach.

Keywords: knowledge based systems, knowledge representation, prudence analysis, ripple-down rules, verification and Validation.

1 Introduction

Brittleness in Knowledge Based Systems (KBS) research has been investigated from many possible angles. For instance: Knowledge Acquisition and Design Structuring (KADS) [1] for extracting deeper knowledge; and, Cyc [2] for capturing general knowledge. A third area of investigation, Verification and Validation (V&V) [3], moved away from finding better methods of acquiring knowledge and instead developed a means of checking whether a KB was complete. V&V, however, is performed with known cases, where the inferenced results can be verified by an expert. Once the system goes online, not all cases can be checked by the expert and any errors by the system go mostly unnoticed. Prudence analysis (PA) extends this by allowing the KBS to detect when an inferenced solution to a case may be wrong. This paper introduces a new approach to PA. The following section will briefly discuss PA, which is followed by a discussion of the approach used in this study. Section 4 will detail the PA experiments and will compare with previous work.

2 Prudence Analysis (PA)

V&V attempts to identify whether all the possible cases are covered by a KBS. PA, however, only uses actual cases as they are presented. Currently, PA has only been studied by a minority of researchers, all of whom have centred their studies on a

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 482–488, 2008.

single family of KBSs, referred to as Ripple-Down Rules (RDR) [4]. The primary reason for this is that RDR is an incremental knowledge acquisition (KA) and maintenance methodology. It is the flexible and maintainable structure of RDR that makes it ideal for PA. Early work on PA, including WISE [5, 6], Feature Recognition Prudence (FRP) and Feature Exception Prudence (FEP) [6-8], were based on comparing the inference case with other paths through the tree or through investigating if any features of the case had not been used for its classification.

[9] took a new approach of comparing cases with previously seen cases within context, and provided warnings if they differed in some unusual way. The method compared individual value-attribute pairs and warned if they exceeded what had been previously seen. At the time of the results in this paper were gathered this was the most recently published study. Therefore, [9] published results are used for comparison with this papers system. The most recent work, however conducted at the same time as this study, was a PhD thesis by [10]. However, a direct comparison with [9]'s published results and this paper work is difficult as a different version of the GARVAN dataset was used. Additionally, cases were removed when the simulation expert could not classify them and missing values were artificially added.

3 Methodology

One problem with the above approaches is the reliance on attributes existence or absence in a case for the generation of warnings. This limit's the methods ability to be applied primarily in domains with a controlled number of only relevant attributes. Domains with large amounts of irrelevant attributes such as free text classification will tend to produce large amounts of false positives. The work in this paper has taken a significantly different approach. Instead of looking at attributes, it analysis the structure of the rule base and the paths followed by the inferencing process. Therefore, the method described in this paper is knowledge driven, rather than the previous approaches which were data driven.

3.1 Multiple Classification Ripple-Down Rules

Ripple-Down Rules is a maintenance centred methodology for a KBS based approach using the concept of fault patching [11] and was first proposed by [4]. It utilises a binary tree as a simple exception structure aimed at partially capturing the context that knowledge is obtained from an expert. The context is the sequence of rules that had evaluated to provide the given conclusion [4, 12-16]. Therefore, if the expert disagrees with a conclusion made by the system they can change it by adding a new rule. The new rule will only fire if the same path of rules is evaluated [14].

Ripple-Down Rules has been shown to be a highly effective tool for knowledge acquisition (KA) and knowledge maintenance (KM). However, it lacks the ability to handle tasks with multiple possible conclusions. Multiple Classification Ripple-Down Rules (MCRDR) aim was to redevelop the RDR methodology to provide a general method of building and maintaining a knowledge base (KB) for multiple classification domains. The methodology developed by [5] is based on the proposed solution by [13, 14]. The primary shift was to switch from the binary tree to an *n-ary* tree representation. Knowledge is acquired by inserting new rules into the MCRDR tree when a

misclassification has occurred. The new rule must allow for the incorrectly classified case, identified by the expert, to be distinguished from the existing stored cases that could reach the new rule [17]. This is accomplished by the user identifying key differences between the current case and each of the rules' cornerstone cases.

3.2 Rated MCRDR

The hybrid methodology used in this paper, referred to as Rated MCRDR (RM), combines MCRDR with an artificial neural network (ANN). This function fitting algorithm learns patterns of fired rules found during the inferencing process. Firstly, a case is presented to the MCRDR tree, which classifies the case. Then for each rule in the inference, an associated input neuron will fire. The network then produces a vector of output values, $\overline{v}$, for the case presented. The system, therefore, provides two separate outputs; the case's classifications and an associated set of values.

Learning in RM is achieved in two ways. Firstly, the value for each corresponding value for $\overline{v}$ receives feedback from the environment concerning its accuracy. The network learns by either using the standard backpropagation approach using a sigmoid thresholding function, and the MCRDR component still acquires knowledge in the usual way. The only exception is when the expert adds a new rule to MCRDR. As the input space grows, new input nodes need to be added to the network in such a way that does not damage already learned information. Therefore, the network structure needed to be altered by adding shortcut connections from any newly created input nodes directly to each output node and using these connections to carry a weight adjustment. When a new input node is added, additional hidden nodes are added.

The *single-step-Δ-initialisation-rule,* Equation 1, directly calculates the required weight for the network to step to the correct solution immediately. This is accomplished by reversing the feedforward process back through the inverse of the symmetric sigmoid. It is possible for the expert to add multiple new rules for the one case. In these situations the calculated weight is divided by the number of new features, *m*. Finally, the equation is multiplied by the step-distance modifier, *Zeta (ζ). Zeta (ζ)* should always be set in the range $0 \le \zeta \le 1$. It allows adjustments to how large a step should be taken for the new features.

$$w_{no} = \zeta \left[\left(\left(\log \left[\frac{f(net)_o + \delta_o + 0.5}{0.5 - (f(net)_o + \delta_o)} \right] \Big/ k \right) - \left[\left(\sum_{i=0}^{n=1} x_i w_{io} \right) + \left(\sum_{h=0}^{q} x_h w_{ho} \right) \right] \right) \Big/ m x_n \right] \quad (1)$$

3.3 RM Applied to Prudence Analysis

The basic idea behind applying RM to PA is to allow MCRDR to develop classifications in the general way, while the network passively watches rules being added to the MCRDR tree. As it watches it also attempts to identify the correct classifications. Through classification testing it was found that the classifications between the MCRDR component and the network often differed when MCRDR misclassified [18]. Therefore, the prudence system developed identifies these differences and warns the user that the classification by MCRDR could be wrong.

Training is a simple process of identifying the correct classification that the expert has agreed to when accepting a case. Obviously, however, this can only be done when

a warning has actually been generated. When no warning is generated the system is unable to train because the system cannot be certain whether the expert would have wanted to alter the classification. When a warning is given and the expert confirms a classification the reward is a positive value at the output where it should have been classified as a particular case and a negative value otherwise.

Thresholding is performed on a per class basis. Basically, if the MCRDR and ANN classes were the same, then a warning was not generated. However, if the network's absolute rating for a particular class was below a certain threshold then it was interpreted as the network being unsure of its rating, and therefore, a warning would be generated. This second method of warning only occurred when the network had the same result as the MCRDR inference engine. This simple tool was found to be highly effective at improving prediction. The thresholding value was also made dynamically adjustable. Therefore, when a warning was warranted, the threshold was increased and when the warning was not needed, the threshold was reduced.

4 Experimental Method

The experiment performed tested the method 10 times with each of the three datasets randomly reordered. The following results are an average of these experiments. This test gathered statistics on how accurate its predictions were. The simulated expert used in this study is similar to those used in other RDR research such as the one used by [9]. C4.5 [19] is used to generate the simulated expert's knowledge base. The resulting tree then classifies each case presented, just like our KB under development. If the KB being constructed, incorrectly classifies a case then the simulated expert's decision tree is used to find attributes within rules that led to the correct classification.

The three dataset used, listed below, are from the University of California Irvine Data Repository [20]. These datasets were used as they are the same as those used by [9]'s study and thereby can be compared.

- CHESS – Using the Chess end game of King+Rook (black) vs King+Pawn (white). This dataset has 36 attributes with a binary classification and 3196 cases.
- TIC-TAC-TOE (TTT) – The complete Tic-Tac-Toe terminating board positions. This dataset has 9 attributes with a binary classification and 958 cases.
- GARVAN – This is a subset of the full dataset that provided medical diagnoses for thyroid problems. Has 29 attributes with 60 classification and 21822 cases.

Table 1. Comparison of the averages between this papers PA system and [9] system

Datasets	Algorithms	False Neg %	True Pos %	False Pos %	True Neg %	Accuracy %
GARVAN	*RM*	0.1	1.7	15	83	92.9
	Compton	0.2	2.4	15	83	91.7
Chess	*RM*	0.2	0.8	8	91	80.0
	Compton	0.3	1.3	7	91	81.3
TTT	*RM*	2.9	8.8	12	76	75.2
	Compton	1.5	3.8	14	81	71.7

4.3 Results

The ability of RM to predict which cases are outside its knowledge base can be measured through a combination of accuracy and the number of warnings. For a prudence system to be viable it requires a high degree of accuracy, while keeping the amount of warnings to a minimum. The level of accuracy can be calculated by dividing the amount of cases that were warned about and for which the expert created rules, by the total amount of cases that needed to have new rules added.

Table 1, shows results averaged over ten randomised runs compared to [9]'s results. It can be seen in these results that RM has outperformed [9] results in two datasets but not in the chess dataset where it performed slightly less. One interesting result was its success on the harder datasets. This indicates that the more complex applications have a greater degree of relationship information available during training. These results indicate that the more real world datasets may still be able to achieve good accuracy even though the little fake domains cause problems.

4.4 Versatility of RM

These results provide an improvement over previous prudence analysis results, indicating the value of RM as a potential predictor of the boundary of knowledge in a KB. While this separation between RM and [9] results is only small, there was one very significant advantage found in the RM approach. The threshold adjusting rate can have a significant and worthwhile affect on the system's results. This parameter can be set to adjust the threshold quickly or slowly. If it adjusts quickly the system is less accurate but produces fewer warnings. If it adjusts slowly then this improves the accuracy at the cost of producing more warnings. Effectively, this allows an expert direct and simple control over the warning and accuracy of the prudence system.

This versatility was noticed during early testing and so a number of tests were performed to verify its occurrence. These tests involved 10 experiments with a different threshold factor and each run with ten different randomisations. Fig 1 shows the results for the GARVAN dataset, where the triangles representing the different performances of the RM approach. The unfilled triangle identifies the result that was

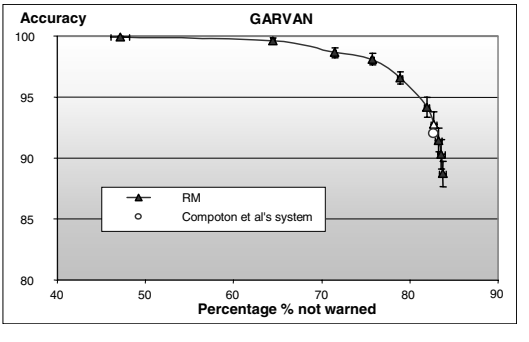


Fig. 1. Compares the results using different threshold adjustment values. The y-axis represents the level of accuracy. The x-axis shows the percentage of cases where a warning was not given.

given in Table 1. The error bars indicate the 95% confidence range for the accuracy and percentage of cases not warned. This chart includes an empty circle which represents the location that [9]'s system achieved.

It can be seen in these results that RM was able to achieve virtually 100% accuracy if warnings are provided on just over 50% of cases. This is clearly a lot of warnings but is a vast improvement on the current use of RDR methodologies where the expert must check every case. What is important is that you can achieve a result very close to perfect with much fewer warnings. The other aspect of these results is that the number of warnings can be reduced to around 84%, in the GARVAN dataset, of cases if the expert can tolerate an accuracy of just fewer than 90%, which is still very high.

5 Conclusion

Prudence analysis represents a method for predicting when a case requires knowledge beyond the system's current KB. It is one way of attempting to resolve the issue of brittleness in current knowledge based systems. In theory, prudence analysis would be a very powerful tool when performing knowledge acquisition and maintenance. Previous work in this area, however, has yielded results that are not sufficiently accurate, or that produce too many warnings, to make such a system viable.

RM studies the internal structure of the KB as it is being developed. The method developed used the classification ability of both hybridised components and compared the results. If they disagreed a warning was produced. The results presented in this paper show that RM is able to predict errors more accurately than previous work without increasing the number of warnings. The most interesting results were those detailing RM's versatility. It could be seen that, through a simple process, the expert could control precisely what level of accuracy was required for the task at hand.

Acknowledgment

The majority of this paper is based on research carried out while affiliated with the Smart Internet Technology Cooperative Research Centre (SITCRC) Bay 8, Suite 9/G12 Australian Technology Park Eveleigh NSW 1430 and the School of Computing, University of Tasmania, Locked Bag 100, Hobart, Tasmania.

References

1. Wielinga, B.J., Schreiber, T.A., Breuker, J.A.: KADS: a modeling approach to knowledge engineering. Knowledge Acquisition 4(1), 5–54 (1992)
2. Lenat, D.B., Guha, R.V., Pittman, K.: Cyc: Toward Programs with Common Sense. Communications of the ACM 33(8), 30–49 (1990)
3. Preece, A.D.: Validation of knowledge-based systems: Current trends and issues. The Knowledge Engineering Review 10(1), 69 (1995)
4. Compton, P., Jansen, R.: Knowledge in Context: a strategy for expert system maintenance. In: Second Australian Joint Artificial Intelligence Conference (AI 1988), vol. 1, pp. 292–306 (1988)

5. Kang, B.: Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules, PhD thesis (1996)
6. Edwards, G.: Reflective Expert Systems in Clinical Pathology (1996)
7. Edwards, G., Kang, B.H., Preston, P.: Prudent expert systems with credentials: Managing the expertise of decision support systems. In: International Journal of Biomedical Computing. RDR prudence and credentiuals is descussed WISE, FRP, FEP, vol. 40, pp. 125–132 (1995)
8. Edwards, G., Compton, P., Kang, B.: New Paradigms for Expert Systems in Healthcare. In: Proceedings of the Fourth Health Informatics of NSW Conference, Primbee, pp. 67–72 (1995)
9. Compton, P., Preston, P.: Knowledge based systems that have some idea of their limits. In: 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW 1999). SRDG publications, Canada (1996)
10. Prayote, A.: Knowledge Based Anomaly Detection (2007)
11. Menzies, T., Debenham, J.: Expert System Maintenance. In: Kent, A., Williams, J.G. (eds.) Encyclopaedia of Computer Science and Technology, vol. 42, pp. 35–54. Marcell Dekker, New York (2000)
12. Beydoun, G.: Incremental Acquisition of Search Control Heuristics, PhD thesis (2000)
13. Compton, P., Edwards, G., Kang, B.: Ripple Down Rules: Possibilities and Limitations. In: 6th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW 1991), vol. 1, pp. 6.1–6.18. SRDG publications, Canada (1991)
14. Compton, P., Kang, B., Preston, P.: Knowledge Acquisition Without Knowledge Analysis. In: Aussenac, N., Boy, G.A., Ganascia, J.-G., Kodratoff, Y., Linster, M., Gaines, B.R. (eds.) EKAW 1993. LNCS, vol. 723, pp. 277–299. Springer, Heidelberg (1993)
15. Preston, P., Edwards, G., Compton, P.: A 1600 Rule Expert System Without Knowledge Engineers. In: Moving Towards Expert Systems Globally in the 21st Century (Proceedings of the Second World Congress on Expert Systems 1993), New York, pp. 220–228 (1993)
16. Preston, P., Edwards, G., Compton, P.: A 2000 Rule Expert System Without a Knowledge Engineer. In: Proceedings of the 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, pp. 17.1–17.10 (1994)
17. Preston, P., Compton, P., Edwards, G.: An Implementation of Multiple Classification Ripple Down Rules. In: Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, University of Calgary, Calgary. Department of Computer Science. SRDG Publications, Canada (1996)
18. Dazeley, R.: To The Knowledge Frontier and Beyond: A Hybrid System for Incremental Contextual-Learning and Prudence Analysis. Uni. of Tasmania, PhD thesis (2007)
19. Quinlan, J.R.: C4.5: Programs for Machine Learning. In: Langley, P. (ed.), p. 302. Morgan Kaufmann Publishers, San Mateo (1993)
20. Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences (1998)

Clustering with XCS on Complex Structure Dataset

Liangdong Shi, Yang Gao, Lei Wu, and Lin Shang

State Key Laboratory for Novel Software Technology, Nanjing University,
No.22, Hankou Road Nanjing, Jiangsu, China
licesh@gmail.com, gaoy@nju.edu.cn,
realvisual@gmail.com, shanglin@nju.edu.cn

Abstract. Learning Classifier System (LCS) is an effective tool to solve classification problems. Clustering with XCS (accuracy-based LCS) is a novel approach proposed recently. In this paper, we revise the framework of XCS, and present a complete framework of clustering with XCS. XCS consists of two major modules: reinforcement learning and genetic algorithm. After the learning process, the learnt rules are always redundant and the large ruleset is incomprehensive. We adopt the revised compact rule algorithm to compress the ruleset, and propose a new rule merging algorithm to merge rules for generating genuine clustering results without knowing of the number of clusters. The experiment results on several complex structure datasets show that out approach performs well on challenging synthetic datasets.

Keywords: XCS, clustering, genetic algorithm, reinforcement learning.

1 Introduction

Learning Classifier System (LCS) has a good performance on supervised learning tasks, especially on classification tasks. Since Wilson proposed XCS and extended it to XCSR to deal with the real-value attributes, LCS methods have become more useful and popular in solving complex real-world data mining problems [1]. Most recent LCS research works are based on XCS or XCSR (for convenience, we always use XCS to denote XCS and XCSR in the following text), such as Gao's LCSE [2][3].

The LCS family can achieve good performance in most situations. But on the other hand, there is a lack of the use of LCS in unsupervised learning tasks. Clustering is one of the most important applications in unsupervised learning technique. A set of data are grouped into clusters without priori knowledge, and the data are similar in the same cluster while dissimilar in different clusters. So far, there have been a variety of clustering algorithms, but no one can works well on all the datasets. These methods can be classified into two categories: (1) according to the clustering criteria, construct a reasonable function and find the extremum, such as K-means; (2) heuristic methods based on the similarity function. The evolution strategies can be treated as heuristic methods.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 489–499, 2008.

Among the evolution strategies, Tseng proposed a genetic approach to automatically clustering [4]. Safaris proposed the NOCEA (Non-Overlapping Clustering with an Evolutionary Algorithm) algorithm, which performed well on challenging datasets [5]. Considering the common strategy of evolution, it is suitable to apply XCS to clustering tasks.

Tamee's approach which attempted to cluster with an accuracy-based LCS was effective and achieved better results compared with k-means [6]. Going through the approach, we find that the way of rule representation is not suitable for clustering results. In simple structure datasets, the distances between data in the same cluster are always smaller than those in different clusters. However, things are different in complex structure datasets. So we using a union set of rules to represent clustering results and apply it on complex structure datasets. Besides, Tamee mainly discussed how to realize the module of reinforcement learning and genetic algorithm in XCS for clustering tasks. He didn't explain the complete architecture in detail. In this paper, we try to give a full and clear presentation of the approach of clustering with XCS. The experiments will show satisfying results on several complex structure datasets in the end.

The paper is organized as follows. In Section 2, we introduce the framework of XCS. In Section 3, we firstly present a revised XCS framework to solve clustering problems, and then state the complete approach in detail. A new rule merging algorithm is also presented to generate genuine clustering results. In Section 4, we conduct some test experiments on the synthetic complex datasets and analyze the results. Finally, we draw some conclusions and future works in Section 5.

2 Introduction to XCS

LCS mainly consists of two important components: (1) reinforcement learning module, for receiving rewards feed back from the environment and updating the parameters of classifier rules; (2) genetic algorithm module, for generating new classifier rules and evolving the rule population. The original LCS has been applied in many fields successfully. However, LCS has many known problems in achieving the near optimal performance, and the framework is very complex and flawed in several aspects, which strongly restrict its application [7].

Thus Wilson proposed a modified version of LCS, which is called XCS. The most important difference between XCS and LCS was the redefinition of fitness. In classical LCS, the fitness of rules was also prediction/strength. By contrast, XCS introduces a new parameter called "accuracy", based on which fitness was calculated [1]. Thus, the original functions of prediction are separated. Prediction now indicates the reward from the environment, and helps to choose actions. The newly introduced fitness denotes the strength of one rule compared to others in population, and is used in genetic algorithm. Another main difference is that the genetic algorithm do not act on the whole population any more, but is executed in niches such as action set or match set.

Fig. 1 shows the framework of XCS. XCS includes two interface modules, a sensor and an effector. The sensor receives an input from the environment

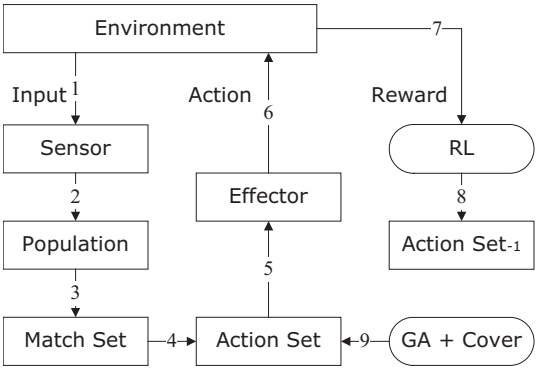


Fig. 1. The framework of XCS

without a class label, and the effector outputs a classification on the input. There are three main data structures in XCS: (1) a population set of rules; (2) a match set and (3) an action set. The population contains all learnt rules. The rules whose conditions match the input data form the match set. In the match set, each rule has an action (or class label). Then a method is used to choose an action. All rules which have the same action form the action set [2].

3 Clustering with XCS

The prediction problem in LCS tasks can be divided into two steps: first we partition the problem space and then get the prediction from the evolutionary partitions. Such definition actually already includes the clustering problem [8].

3.1 Framework for Clustering

Although the framework for clustering is similar to XCS, there are several components needed to be revised to fit clustering problems. (1) For lacking of feedback mechanism, the module of action set is no longer needed. Accordingly, reward is not returned by the environment, but is calculated as reciprocal of the Euclidean distance between the centre of a rule and the input data from the sensor; (2) Rule compaction and merging module are added to generate genuine clustering results. The following Fig. 2 shows the framework. In Table 1, we state the algorithm further.

Clustering based on XCS mainly consists of two steps. First is the learning process, after which we get a ruleset P. This ruleset covers the whole dataset S, and $|P| < |S|$. Then we compact the ruleset and merge the rules to get a simple and comprehensible one called C. Still it covers S, and $|C| < |P|$. Now C can be treated as the clustering results. More details will be given in the following subsections.

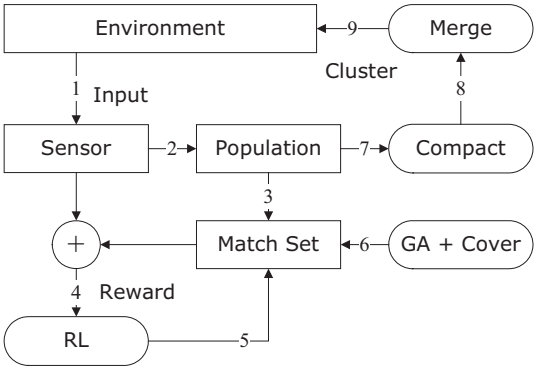


Fig. 2. The framework of clustering with XCS

Table 1. The algorithm of clustering with XCS

1. Create the population of size N in XCS and initialize it.
2. while not reach the maximum learning step T {
2.1 Get an input from the environment, and scan the whole population to form the match set.
2.2 If there's no rule match the input, using "Covering" to generate a new rule and insert into the population.
2.3 Calculate the reward of each rule in the match set, and update parameters using the rewards.
2.4 If needed, execute genetic algorithm on the match set.}
3. Apply the revised ruleset compact algorithm on the evolved population P.
4. Merge the rules in the compressed ruleset to get clustering results C.

3.2 Population Initialization

We adopt the way of rule representation used in XCSR to process continuous values. In XCSR, Wilson changed the original rule representation structure and used the interval representation [9]. A continuous-valued attribute is represented as an interval in the form of:

$$interval_i = (c_i, s_i) \quad c_i, s_i \in R;\ i = 1, 2, ...d \tag{1}$$

c_i is the interval centre and s_i is the spread from c_i and d is the number of dimensions or attributes. Then each interval is calculated as $[c_i\text{-}s_i,\ c_i\text{+}s_i]$.

The maximum size of the population is fixed as N. There're a variety of ways to initialize it. Here we simply choose the random way to generate N_0 rules. N_0 is less than N with the purpose of supplying some elasticity for generic algorithm use. The centres of intervals is randomly distributed all over the sample space.

The spread of each rule in dimension i is randomly generated in $(0, s_0)$, where s_0 is the maximum spread predefined in the learning process.

Once the sensor gets an input from the environment, each rule is examined and tagged as a member of match set if it is contained in $[c_i\text{-}s_i, c_i\text{+}s_i]$ for each dimension i. If no rule matches the input, "covering" is activated to generate a rule and add the rule into the population. The rule's centre is the input, and spread is calculated in the same way mentioned above.

3.3 Adjust Rule Parameter

In XCS, fitness is calculated based on accuracy. Although this method was explored for classification, it is also suitable for clustering problems because of its generalization capabilities. We will introduce the method used in XCS here. Each rule is associated with prediction (p), prediction error (ϵ), and fitness (f) parameters. Reinforcement learning is applied to update these parameters [1].

The standard Widrow-Hoff delta rule is used here to calculate p, ϵ and f. First, accuracy κ_j is defined as a function of ϵ:

$$\kappa_j = \begin{cases} \exp \frac{ln(\alpha)(\epsilon_j - \epsilon_0)}{\epsilon_0} & if \epsilon_j > \epsilon_0, \\ 1 & if \epsilon_j \leq \epsilon_0. \end{cases} \tag{2}$$

Then calculate the relative accuracy $\kappa'_j = \kappa_j / \Sigma \kappa_j$. Thus fitness can be adjusted as:

$$f_j \leftarrow f_j + \beta(\kappa_j - f_j) \tag{3}$$

As mentioned in section 3.1, the reward is no longer the feedback from the environment. Here, we define it as reciprocal of the Euclidean distance between the center of a rule and the input data from the sensor:

$$R = \frac{(\sum_{i=1}^{d}(x_i - c_{ij})^2)^{1/2} + 2 \times \rho}{(\sum_{i=1}^{d}(x_i - c_{ij})^2)^{1/2} + \rho} \tag{4}$$

ρ is to prevent dividing by 0. We set it to 1 in our experiments. So, the prediction error is adjusted toward the absolute difference $|R$ - $p_j|$:

$$\epsilon_j \leftarrow \epsilon_j + \beta(|R - p_j| - \epsilon_j) \tag{5}$$

Finally, prediction can be adjusted as:

$$p_j \leftarrow p_j + \beta(R - p_j) \tag{6}$$

α, ϵ_0, β are predefined parameters. Moreover, the MAM technique is used [1]. That means the parameters are set to the average value before $1/\beta$ learning times. The benefit is making the system less sensitive to the randomly initialized population and adjusting the parameter values more quickly.

3.4 Evolve Population

Genetic Algorithm acts on the match set to evolve rules in population. There is an approach to control the GA running times to improve the performance [1]. Each rule maintains a time stamp, which is the last time GA effected on the rule. GA is applied in this match set only when the average number of time steps since the last GA is greater than θ_{GA}.

GA is executed as follows: First, two parents are chosen according to their fitness and prediction; Second, two offspring are produced through crossing and mutation; At last the two offspring are inserted into the population. If the population is full, we must select one to remove. The smallest prediction or the smallest fitness or the longest last used rule can be chosen to delete.

The two offspring cross in the way that any spread in dimension i has the probability χ to change with each other. Mutation is used to adjust the spread size. We realize it by randomly picking a value from [-0.01, 0.01], and adding it to the original spread. The mutation probability in each dimension is μ.

3.5 Compact Ruleset

When the learning process is finished, the ruleset is redundant. Wilson proposed the CRA (Compact Rule Algorithm) to reduce the size of evolved population without loss of performance [10]. The main idea is to select accurate and generalized rules from the mutual population. Here we realize rule compaction in clustering problems based on Gao's aCRA (amendatory Compact Ruleset Algorithm) [2]. It is shown in Table 2, f_0 and p_0 are predefined values.

Table 2. Compact ruleset algorithm for clustering

1. Remove the ith rule from the population set which meet $f_i < f_0$ and $p_i < p_0$.
2. Order the remaining rules based on the product of prediction and fitness.
3. Determine whether or not to add the rule in the top into final compacted ruleset{
3.1. Computer the number n of inputs matched with respect to the train data set.
3.2. Add the rule into final compacted ruleset if n is greater than threshold.
3.3. Remove the matched instances from the train data set.}
4. Terminate if the train data set is empty, else go to step 3.

3.6 Rule Merging

After compacting ruleset, there are still lots of rules in population. These learnt rules overlap and don't form a division of the problem space. So it is important to merge the learnt rules to get genuine clustering results. We will introduce a

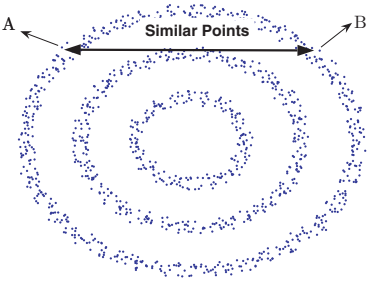


Fig. 3. The well separated complex structure dataset of three-circle

Table 3. Rule merging algorithm for clustering

1. Order the rules based on the product of prediction and fitness.
2. Pick out the top rule and generate a union set U_i containing the rule, then remove it from the population.
3. Scan all the remaining rules, if the rule overlap any one in U_i and satisfy the condition that the overlapped area match at least one sample data, add it into U_i and remove it from the population.
4. If no rule be added into U_i in step 3, go to step 5; else go back to step 3.
5. Go to step 2 if the remaining population is not empty.
6. Check the union sets, if the size of U_i is smaller than threshold, merge it into the nearest union set according to Euclidean distance.

rule merging algorithm for clustering. At first, we should notice that the original way of rule representation doesn't suit for clustering results. Such representation performs well on simple datasets in which similarity function is based on Euclidean distance. But in complex structure datasets, similar data points always far from each other scaled by Euclidean distance. The dataset is shown in Fig. 3, the "distance" between point A and B is long, but they belong to the same cluster.

Here we simply use a set of original rules to represent a cluster. That is, a sample data belongs to a cluster if and only if it matches any one of the rules in the union set. Therefore, we give the merging algorithm in Table 3:

4 Experimental Results

In this section, we present experiment results of clustering with XCS on several datasets. In Table 4, the values of parameters used in these experiments are listed. S_0,ϵ_0,f_0 and p_0 are based on the experimental results. The other parameters are same as the parameters used in XCS [1]. Details of these experiments are discussed below.

We first carry out experiments on the challenging dataset mentioned in Fig. 3. The synthetic dataset has 3 randomly generated circles in $d = 2$ dimension, each

Table 4. Parameter Settings

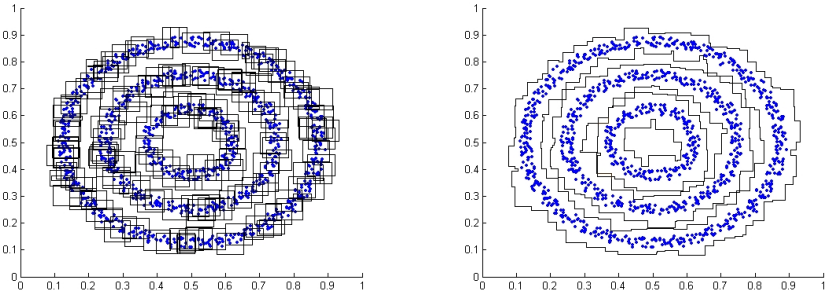
Parameter	Description	Value
N	Size of population	1000
N_0	Size of initial population	800
T	Total learning steps	2000-20000
S_0	Parameter of creating random spread of a rule	0.04
α	Parameter of computing accuracy of a rule	0.1
ϵ_0	Threshold of computing error of a rule	0.03
β	Learning rate of reinforcement learning	0.2
θ_{GA}	The number steps when activating GA process	12
χ	Probability of cross operator	0.8
μ	Probability of mutation operator	0.04
f_0	Initial value and threshold of fitness of a rule	0.05
p_0	Initial value and threshold of prediction of a rule	1

circle has 400 data points. There's a parameter w used to control the width of each cluster. Here $w = 0.15$.

After the learning process and compacting ruleset, the remaining rules are shown in Fig. 4(a). As can be seen, rules overlap much in the same cluster, but are separated among different clusters. Then we apply the rule merging algorithm on it, the final clustering result is shown in Fig. 4(b). From the result, we can find that the testing dataset are divided into 3 clusters quite clearly. Although the rules in the final results still overlap somewhere, the merging algorithm ensures that no sample data exists in this overlapped area. So the clustering results is clear all the same.

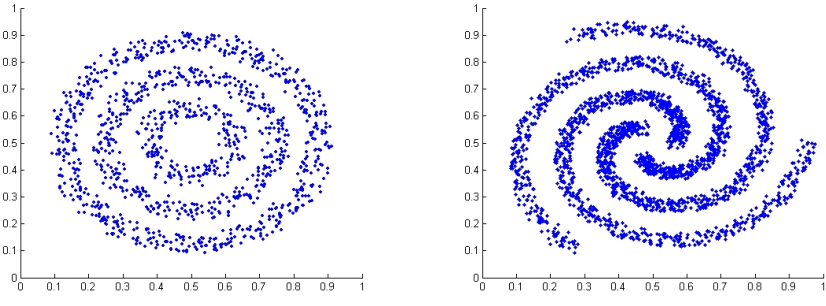
To test the performance of our system further, we increase the complexity by changing the parameter of width w to 0.3. It is shown in Fig. 5(a). The data points become more separated and there are several points between clusters, which increase the difficulty of clustering. Another synthetic dataset with $w = 0.4$ is given in Fig. 5(b). There are 3 spiral arms in this dataset, each has 800 data points. Experiments on this dataset are harder because of the more complex spiral structure. Fig. 6 shows the final results. We note that a few data points are not included in any rules. In this case, we only need to find out the nearest rule and join it into the corresponding cluster.

From the results, we find that the rules cover almost every data points after the XCS learning process and rule compaction. Further more, most of the rules in the same cluster are overlapped while in different clusters are separated. Then all the rules are correctly merged into 3 clusters according to the rule merging algorithm. By contrast, many traditional algorithms of clustering failed on these



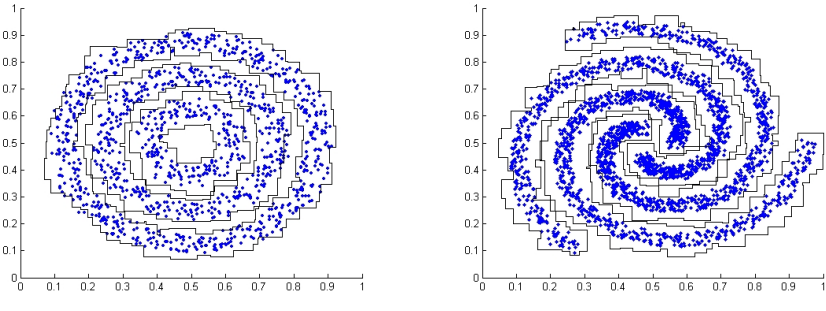
(a) (b)

Fig. 4. After XCS learning process and compacting rules, the clustering results before rule merging (a) and after rule merging (b) on the well separated three-circle dataset



(a) (b)

Fig. 5. Two more complex datasets which are less separated: (a) three-circle and (b) spiral



(a) (b)

Fig. 6. The final clustering results for three-circle (a) and spiral (b) dataset

complex structure datasets, including the well known K-means [11]. On the other hand, our approach is not suitable for this issue that different clusters are fuzzy where the fuzzy data points are near each other. In this case, the rule merging condition which is based on density can be well suited, and it has the ability to solve cases with lots of outliers or noise [5]. Nevertheless, the datasets must be very large to ensure the performance.

5 Conclusions and Future Work

In this paper, we present a complete clustering system using XCS. First, the randomly initialized population are evolved after the revised XCS for clustering. These learnt rules are well generalized after the learning process. Second, the compact rule algorithm is used to reduce the total population size without loss of performance. At last, all the remaining rules are merged according to the newly proposed rule merging algorithm. The merged rules form the genuine clustering results without prior knowledge of the number of clusters. The experiments on several synthetic complex structure datasets show satisfactory clustering results and reliable performance. The learning process of XCS is hard to comprehend. In the future, we will try to make the learnt results more interpretative. The learnt rules are unstable. We are also working on how to make it have less effect on the rule merging process. The parameter settings are still needed to be explored to achieve better performance. Besides, we will apply the system to real world datasets.

Acknowledgments. This paper is supported by the Natural Science Foundation of China (No.60775046 and No.60721002) and the Jiangsu Province Science and Technology(BE2006011).

References

1. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation 3(2), 149–175 (1995)
2. Gao, Y., Wu, L., Huang, J.Z.: Learning classifier system ensemble and compact rule set. Connection Science 19(4), 321–337 (2007)
3. Gao, Y., Huang, J., Rong, H., Gu, D.: Learning classifier system ensemble for data mining. In: Proceeding of Genetic and Evolutionary Computation Conference, pp. 63–66 (2005)
4. Tseng, L.Y., Yang, S.B.: A genetic approach to the automatic clustering problem. Pattern Recognition 34(2), 415–424 (2001)
5. Sarafis, I., Trinder, P., Zalzala, A.: Mining comprehensible clustering rules with an evolutionary algorithm. In: Proceeding of Genetic and Evolutionary Computation Conference, pp. 2301–2312 (2003)
6. Tamee, K., Bull, L., Pinngern, O.: Towards clustering with xcs. In: Proceeding of Genetic and Evolutionary Computation Conference, pp. 1854–1860 (2007)
7. Wilson, S.W., Goldberg, D.E.: A critical review of classifier systems. In: Proceedings of the Third International Conference on Genetic Algorithms, pp. 244–255 (1989)

8. Butz, M.V.: Tutorial of learning classifier systems. In: Proceeding of Genetic and Evolutionary Computation Conference (2007)
9. Wilson, S.W.: Get real! xcs with continuous-valued inputs. In: Lanzi, et al. (eds.) pp. 209–219 (2000)
10. Wilson, S.W.: Compact rulesets from XCSI. In: Proceedings of the 4th International Workshop on Learning Classifier Systems, pp. 197–210 (2002)
11. Ding, J., Chen, S., Ma, R., Wang, B.: A fast directed tree based neighborhood clustering algorithm for image segmentation. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4234, pp. 369–378. Springer, Heidelberg (2006)

A Generalized Joint Inference Approach for Citation Matching

Zhihua Liao[1] and Zili Zhang[1,2]

[1] Intelligent Software and Software Engineering Laboratory
Southwest University, Chongqing 400715, China
[2] School of Engineering and Information Technology
Deakin University, Geelong, VIC 3217, Australia

Abstract. Citation matching is the problem of extracting bibliographic records from citation lists in technical papers, and merging records that represent the same publication. Generally, there are three types of datasets in citation matching, i.e., sparse, dense and hybrid types. Typical approaches for citation matching are Joint Segmentation (Jnt-Seg) and Joint Segmentation Entity Resolution (Jnt-Seg-ER). Jnt-Seg method is effective at processing sparse type datasets, but often produces many errors when applied to dense type datasets. On the contrary, Jnt-Seg-ER method is good at dealing with dense type datasets, but insufficient when sparse type datasets are presented. In this paper we propose an alternative *joint inference* approach–Generalized Joint Segmentation (Generalized-Jnt-Seg). It can effectively deal with the situation when the dataset type is unknown. Especially, in hybrid type datasets analysis there is often no a priori information for choosing Jnt-Seg method or Jnt-Seg-ER method to process segmentation and entity resolution. Both methods may produce many errors. Fortunately, our method can effectively avoid error of segmentation and produce well field boundaries. Experimental results on both types of citation datasets show that our method outperforms many alternative approaches for citation matching.

1 Introduction

Citation matching is the problem of extracting bibliographic records from citation lists in technical papers, and merging records that represent the same publication [3]. It is the problem currently handled by systems such as CiteSeer [1]. Such systems process a large number of scientific publications and extract their citation lists. By grouping together all coreferring citations, the system constructs a database of "paper" entities linked by their relations. A key aspect of this problem is to determine when two observations describe the same object. Citation matching has been a major focus of information extraction field.

Citation matching approach usually consists of *non-joint inference* and *joint inference*. Non-joint inference refers to the process that information extraction and coreference resolution are performed as two independent steps; while joint inference means that information extraction and coreference resolution are performed together in a single integrated inference process.

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 601–607, 2008.

Traditional method is *non-joint inference*. With the progress of probabilistic inference and machine learning, *joint inference* has been taken up and applied to practice. Currently, research focus is *joint inference*. For example, Pasula et al.[2] developed a "collective" model in which the segmentation was performed in the pre-processing stage. Wellner et al. [10] extended the pipeline model by passing uncertainty from the segmentation step to the entity resolution step, but did not "close the loop" by repeatedly propagating information in both directions. Poon and Domingos [3] recently developed a full *joint inference* model which consist of two components: Segmentation step and Entity Resolution step. In segmentation step, they introduced two methods: Joint Segmentation (Jnt-Seg) and Joint Segmentation Entity Resolution (Jnt-Seg-ER). Essentially, citation datasets can be classified as sparse type (in citations the similar title-like strings do not occur both at boundaries and inside fields), dense type (for citations the similar title-like strings occur both at boundaries and inside fields) and hybrid type. Generally, Jnt-Seg method is effective at processing sparse type datasets, but often produces many errors when applied to dense type datasets. On the contrary, Jnt-Seg-ER method is good at processing dense type datasets, but insufficient when sparse type datasets are presented. However, when the hybrid data type is presented both methods may produce unsatisfactory results.

With such observations in mind, we extended Jnt-Seg approach to a more generalized joint segmentation approach, namely Generalized-Jnt-Seg. The advantage of this approach is that regardless which type of the input dataset is, it can perform full *joint inference* effectively, and improve the segmentation accuracy. In addition, the computational expenses are lower compared with Jnt-Seg.

The rest of the paper is organized as follows: Jnt-Seg and Jnt-Seg-ER models proposed by Poon and Domingos are described in Section 2. Section 3 presents our proposed Generalized-Jnt-Seg model. Section 4 presents the experimental results and Section 5 concludes the paper.

2 Poon-Domingos Models

In citation matching, two widely used approaches are Jnt-Seg model and Jnt-Seg-ER model introduced by Poon and Domingos [3]. In order to present our method, we firstly introduce the basic ideas of these two models. As nested Markov logic networks are used in citation matching, a brief introduction to Nested Markov Logic network is provided.

Nested Markov logic network (nMLNs) [4] is usually used to address the problems such as the asymmetry treatment between conjunctions and disjunctions, and between universal and existential quantifiers caused by standard Markov logic networks (MLNs). Such problems are common in citation matching. Markov logic is a probabilistic extension of finite first-order logic [5]. It combines first-order logic with Markov networks. While MLNs soften conjunctions and universals, disjunctions and existential quantifiers remain deterministic. This can be overcome by allowing the features of MLNs to be nMLNs. This representation

can also be called recursive random fields (RRFs) [4]. It enables RRFs to better capture features through weight learning.

The main evidence predicates in the MLNs are $Token(t, i, c)$ and $HasPunc(c, i)$. The query predicates are $InField(i, f, c)$ and $SameCitation(c, c')$. After given all predicates, MLNs for entity resolution (ER) can be constructed [7]. To reduce the excessive segmentation noise, two predicates are defined and used to solve this problem, namely $SimilarTitle(c, i, j, c', i', j')$ and $SimilarVenue(c, c')$. Hence the simple rule is formed as follow:

$$\text{SimilarTitle}(c, i, j, c', i', j') \wedge \text{SimilarVenue}(c, c') \Rightarrow \text{SameCitation}(c, c') \qquad (ER)$$

This simple MLNs can effectively carry out entity resolution in citation matching.

Followed by MLNs constructing for ER, the joint segmentation model can be formed. In citation matching, since segmenting a citation can help segment similar ones, this information can be incorporated by defining one predicate called $JntInfCan(c, i, c')$ After adding this predicate, the joint segmentation model is represented by both rules of the form:

$$\text{InField}(i, +f, c) \wedge \neg\text{HasPunc}(c, i) \wedge (\neg\exists c' \text{JntInfCan}(c, i, c')) \Rightarrow \text{InField}(i+1, +f, c) \qquad \text{(Jnt-Seg)}$$

$$\text{InField}(i, +f, c) \wedge \neg\text{HasPunc}(c, i) \wedge (\neg\exists c' \text{JntInfCan}(c, i, c') \wedge \text{SameCitation}(c, c')) \Rightarrow \text{InField}(i+1, +f, c) \qquad \text{(Jnt-Seg-ER)}$$

Here the first rule is called Jnt-Seg method and the second rule called Jnt-Seg-ER method.

In practice, we commonly use Jnt-Seg method to deal with sparse type data and take Jnt-Seg-ER method to handle dense type data. However, it seems to be very difficult to choose suitable method to process the hybrid type data. Here we illustrate such situations with following examples. As can be seen, citations in example (i) and example (ii) are probably referring to the same paper, despite many superficial differences.

(i)
- Minton, S(1993 b). Integrating heuristics for constraint satisfaction problems: A case study.In: Proceedings AAAI.
- S. Minton Integrating heuristics for constraint satisfaction problems: A case study. In AAAI Proceedings,1993.

(ii)
- Robert E. Schapire. 5(2) The strength of weak learn ability. Machine Learning, 1989 197-227.
- R. Schapire. On the strength of weak learn ability. Proceedings of the 30th I.E.E.E. Symposium on the Foundations of Computer Science, 1989, pp. 28-33.

In (i), the citations are sparse type; and citations represent dense type in (ii). For the first citation of (i) and (ii), author and title are clearly separated by a date and period, and extracting then is fairly straightforward. But there is no clear author-title boundary in the second citation of (i) and (ii), and correctly

segmenting them seems very difficult. In such situations, Jnt-Seg method can be applied to correctly segment the second citation of (i), but for second citation of (ii), the Jnt-Seg method will produce erroneous boundary. The reason is that Jnt-Seg method will erroneous segment the substring of title "On the strength of" into two substrings "On" and "the strength of". To overcome this problem, Jnt-Seg-ER method was proposed. It can correctly handle the second citation of (ii), but it can not correctly segment the second citation of (i). Because the predicate $SameCitation()$ in the Jnt-Seg-ER rule will erroneous segment the name of the author "S. Minton" into "S" and "Minton" and treat the second substring as a part of title.

3 The Generalized-Jnt-Seg Model

In consideration of the shortcomings of Jnt-Seg and Jnt-Seg-ER methods, we want to find an alternative to effectively solve this problem. Our method is to add the predicates $JntInfCan(c, i+j, c'), j = 0, 1, 2, \cdots, m$ to the Jnt-Seg rule. By adding such predicates, we extend the Jnt-Seg rule to handle dense type citations. Thus, the extended rule can process sparse type, dense type as well as hybrid type.

Considering such situations, in some cases the substring of title in the first citation may start to match the substring of title in the second similar title-like citation from the second word. We can add a predicate $JntInfCan(c, i+1, c')$ in Jnt-Seg rules to shift the matching and avoid erroneous segmentation. Thereby we form following rules:

$$\text{InField}(i, +f, c) \wedge \neg\text{HasPunc}(c, i) \wedge (\neg\exists c' \text{JntInfCan}(c, i, c') \vee \text{JntInfCan}(c, i+1, c')) \Rightarrow \text{InField}(i+1, +f, c) \quad \text{(Ged-Jnt-Seg(j=1))}$$

Likewise, when the successive two words of the substring of title in the first citation can not match with the substring of title in the second citation, but begin to match from the third words, we can add the predicate $JntInfCan(c, i+2, c')$, and joint it to form such rules as follow:

$$\text{InField}(i, +f, c) \wedge \neg\text{HasPunc}(c, i) \wedge (\neg\exists c' \text{JntInfCan}(c, i, c') \vee \text{JntInfCan}(c, i+1, c') \vee \text{JntInfCan}(c, i+2, c')) \Rightarrow \text{InField}(i+1, +f, c) \quad \text{(Ged-Jnt-Seg(j=2))}$$

In general, we can introduce the predicate $JntInfCan(c, i+j, c')$ and joint it. Here j is a positive integer and can be set to 1, 2, 3 and so on. As a result, we can generalize above rules and get more general rules as follows:

$$\text{InField}(i, +f, c) \wedge \neg\text{HasPunc}(c, i) \wedge (\neg\exists c' \text{JntInfCan}(c, i, c') \vee \text{JntInfCan}(c, i+1, c') \vee \cdots \vee \text{JntInfCan}(c, i+j, c')) \Rightarrow \text{InField}(i+1, +f, c) \quad (m = Z, j = 1, 2, 3, \cdots, m.) \quad \text{(Ged-Jnt-Seg(j=m))}$$

whether a bigger value of j gives a better performance will be answered in Section 5.

One problem of our Generalized-Jnt-Seg method is that it includes many disjunctions and existential quantifiers. However, the MLNs can not handle disjunctions and existential quantifiers effectively. Often, considerable time and computational expenses are needed. We attempt to address this drawback by converting MLNs to RRFs which can easily soften disjunctions and existential quantifiers. Every MLNs can be converted into a relational RRF by translating each clause into an equivalent parfeature.

4 Experimental Results

The standard CiteSeer and Cora datasets are employed to conduct the experiments. The CiteSeer dataset[1] is of sparse type and the Cora dataset[10] is of dense type. In CiteSeer, we randomly split citations into four subsets for four-fold cross-validation; and for Cora, we split them into three subsets for three-fold cross-validation. Firstly, we divided the citation into author field, title field and venue field. Then we carried out standard normalization, simple stemming, and hyphen removal for preprocess. Secondly, we implemented weight learning and inference. For MLNs we used the open source Alchemy (`http://alchemy.cs.washington.edu`). We first implemented the inference in MLNs using a "slice sampling" Markov Chain Monte Carlo algorithm (MC-SAT) [8], and weights learning using voted perceptron [6,9]. Then, we performed inference in RRFs using Markov Chain Monte Carlo (MCMC) and Iterated Conditional Modes (ICM), and weight learning using a novel variant of the back propagation (BP) algorithm [4]. For segmentation, we measured F1 for InField, which was the harmonic mean of recall and precision. For entity resolution in CiteSeer, we measured cluster recall;in Cora we measured both cluster recall and pairwise recall/precision.

The result from Table 1 indicates that MLNs can not effectively handle the disjunctions and existential quantifiers. The memory and time for MLNs grow rapidly with the number of objects. Especially, when the value of j is larger than 3, the memory runs out in the analysis of Citeseer and Cora datasets. On the contrary, RRFs are more efficient in handling the disjunctions and existential quantifiers. Since RRFs and MLNs produce the same results and RRFs are more effective for handling the disjunctions and existential quantifiers, we just report the results obtained by using RRFs in Table 2.

Table 2 compares Poon-Domingos methods with our method in joint segmentation. The "Total" column shows results on all $InField$ atoms, and the remaining columns show results for each field type. The "All" section denotes the results of all citations, "Nontrivial" denotes non-singleton citations, and "Potential" denotes citations with poor author-title boundary. The results for Generalized-Jnt-Seg method are reported by using the values of j from 1 to 4. Since larger value of j requires more computational resources and time, $j = 4$ is used as the optimal trade-off between segmentation accuracy and computational expenses. For Citerseer, our methods clearly outperform the Jnt-Seg method. For Cora, our methods also slightly outperform the Jnt-Seg-ER method.

Table 1. MLNs vs. RRFs on memory and time

Ged-Jnt-Seg		Memory/bytes	Time/seconds
MLNs	$j = 1$	$2.7 * 10^6$	$1.47 * 10^5$
	$j = 2$	$3.5 * 10^6$	$2.23 * 10^5$
RRFs	$j = 1$	$1.2 * 10^6$	$0.58 * 10^5$
	$j = 2$	$1.7 * 10^6$	$0.79 * 10^5$

Table 2. Joint segmentation using different methods with CiteSeer and Cora datasets

	CiteSeer segmentation F1 on each section				Cora segmentation F1 on each section			
All	Total	Author	Title	Venue	Total	Author	Title	Venue
Jnt-Seg-ER	94.50	94.80	93.00	95.30	98.40	99.50	97.60	98.30
Jnt-Seg	94.50	94.90	93.00	95.30	98.40	99.50	97.50	98.30
Ged-Jnt-Seg ($j = 1$)	94.52	94.91	93.02	95.31	98.42	99.52	97.55	98.31
Ged-Jnt-Seg ($j = 2$)	94.52	94.92	93.03	95.31	98.43	99.53	97.58	98.31
Ged-Jnt-Seg ($j = 3$)	94.53	94.92	93.03	95.31	98.43	99.53	97.59	98.31
Ged-Jnt-Seg ($j = 4$)	94.53	94.92	93.03	95.31	98.44	99.53	97.60	98.31
Nontrivial	Total	Author	Title	Venue	Total	Author	Title	Venue
Jnt-Seg-ER	94.80	95.10	93.30	95.60	98.50	99.50	97.70	98.30
Jnt-Seg	94.90	95.20	93.60	95.70	98.50	99.50	97.70	98.30
Ged-Jnt-Seg ($j = 1$)	94.91	95.21	93.61	95.70	98.51	99.51	97.71	98.30
Ged-Jnt-Seg ($j = 2$)	94.91	95.22	93.61	95.70	98.51	99.51	97.71	98.30
Ged-Jnt-Seg ($j = 3$)	94.92	95.22	93.61	95.70	98.51	99.51	97.71	98.30
Ged-Jnt-Seg ($j = 4$)	94.92	95.22	93.61	95.70	98.51	99.51	97.71	98.30
Potential	Total	Author	Title	Venue	Total	Author	Title	Venue
Jnt-Seg-ER	93.90	94.40	92.00	94.90	98.90	99.30	97.90	99.00
Jnt-Seg	94.30	94.50	92.40	95.30	98.30	99.30	97.90	99.00
Ged-Jnt-Seg ($j = 1$)	94.31	94.51	92.42	95.30	98.31	99.32	97.93	99.00
Ged-Jnt-Seg ($j = 2$)	94.32	94.51	92.43	95.31	98.32	99.33	97.95	99.00
Ged-Jnt-Seg ($j = 3$)	94.32	94.51	92.43	95.31	98.32	99.33	97.96	99.00
Ged-Jnt-Seg ($j = 4$)	94.42	94.51	92.43	95.31	98.42	99.33	97.96	99.00

Table 3. Entity resolution using different methods with CiteSeer and Cora datasets

	CiteSeer entity resolution: cluster recall on each section				Cora entity resolution: pairwise recall/precision and cluster recall	
Approach	Constraint	Face	Reason	Reinforce	Pairwise Rec./Prec	Cluster Recall
Fellegi-Sunter [11]	84.3	81.4	71.3	50.6	78.0/97.7	62.7
Lawrence [1]	89	94	86	79		
Pasula [2]	93	97	96	94		
Wellner [10]	95.1	96.9	93.7	94.7		
Poon-Domingos [3]	96.0	97.1	95.1	96.7	94.3/97.0	78.1
Ged-Jnt-Seg($j = 4$)	96.7	97.6	95.4	97.2	94.5/96.9	78.5

Table 3 shows that our approach outperforms many alternatives in entity resolution using CiteSeer and Cora datasets. The results for Fellegi-Sunter [11], Lawrence [1], Pasula [2], Wellner [10], and Poon-Domingos [3] are taken from the corresponding papers. For CiteSeer, the result reaches to 96.7 on Constraint, 97.6 on Face, 95.4 on Reason and 97.2 on Reinforce. The result of Cora, in pairwise recall/precision and cluster recall is 94.5/96.9 and 78.5, respectively.

5 Conclusion

In this paper, we proposed a generalized joint inference method, which is the extension of joint segmentation method. Our approach can effectively process any datasets - sparse, dense, or hydrid, and avoid error of segmentation which in turn improves the accuracy in citation matching. Moreover, by extending MLNs to RRFs, joint inference can be accomplished more efficient. Experimental results on two benchmark datasets show that our approach can effectively avoid error of segmentation and achieve higher accuracy than many alternative approaches. We will conduct more experiments in the future to evaluate the impact of the value j in Generalized-Jnt-Seg ($j = m$) model on the segmentation accuracy and the computational expenses to find an optimal trade-off between accuracy and efficiency.

References

1. Lawrence, S., Bollacker, K., Giles, C.L.: Autonomous citation matching. In: Proc. International Conference on Autonomous Agents, pp. 392–393 (1999)
2. Pasula, H., Marthi, B., Milch, B., Russell, S., Shpitser, I.: Identity uncertainty and citation matching. In: Proc. NIPS 2003, pp. 1425–1432 (2003)
3. Poon, H., Domingos, P.: Joint inference in information extraction. In: Proc. AAAI 2005, pp. 913–918 (2007)
4. Lowd, D., Domingos, P.: Recursive Random Fields. In: IJCAI, pp. 950–955 (2007)
5. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62, 107–136 (2006)
6. Singla, P., Domingos, P.: Discriminative training of Markov logic networks. In: Proc. AAAI 2005, pp. 868–873 (2005)
7. Singla, P., Domingos, P.: Entity resolution with Markov logic. In: Proc. ICDM 2006, pp. 572–582 (2006)
8. Singla, P., Domingos, P.: Memory-Efficient Inference in Relational Domain. In: Proc. AAAI 2006 (2006)
9. Lowd, D., Domingos, P.: Efficient Weight Learning for Markov Logic Networks. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 200–211. Springer, Heidelberg (2007)
10. Wellner, B., McCallum, A., Peng, F., Hay, M.: An integrated, conditional model of information extraction and coreference with application to citation matching. In: Proc. UAI 2004, pp. 593–601 (2004)
11. Fellegi, I., Sunter, A.: A theory for record linkage. J. American Statistical Association 64, 1183–1210 (1969)

Agent-Based Collaborative System and Case-Based Conflict Resolution Process in Preliminary Ship Design

Kyung Ho Lee[1], Jae Joon Lee[2], Young Soo Han[2], Jung Min Lee[2], and Byung Hak Lee[2]

[1] Department of Naval Architect & Ocean Engineering, INHA University, Incheon, Korea
kyungho@inha.ac.kr
[2] INHA University, Graduate School

Abstract. Recently, the shipbuilding companies are assigning design and production environments around the world in different areas. Therefore, the concept of simultaneous engineering and concurrent design becomes very significant. In this paper, a basic architecture of an agent system is proposed to support the exchange and sharing of design information by means of ACL (Agent Communication Language) in preliminary ship design. Above all, the conflicts that occur in the middle of knowledge sharing in the system must be resolved. One approach is to adopt a case-based conflict resolution strategy formulated to resolve current conflict on the basis of previous resolved similar cases in agent-based collaborative design system environments. In addition, conflict resolution handler located in the facilitator which is an agent to control other sub-agents, is developed to treat conflict problems effectively.

1 Introduction

The importance of the design methodology based on the concurrent engineering in distributed environment is increasing from day to day. Many attempts have been made to resolve the limitations of communication between distributed and heterogeneous systems and the sharing of the design information with each other. The adoption of agent technology is one of the solutions to solve these problems [1,2]. Many attempts have been made to construct collaborative systems in order to support cooperation in distributed ship design environments [3,4]. As ship designing is a very complicated process and manipulates much data, it is necessary to construct the collaborative system based on the agent technology. In this paper, the implemented agent-based system for preliminary ship design is presented. In addition, this paper focuses on the methodology of conflict resolution based on Case-Based Reasoning (CBR) to resolve the conflicts occurred in the agent-based ship design system.

2 Ship Design and Multi-agent System

In Korean shipyards, each field of design utilizes different CAD systems such as TRIBON system for the preliminary design, SIKOB in the hydrostatic calculation, CADRA or AutoCAD in sketch designing, AUTODEF, TRIBON, or the Intergraph

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 608 – 614, 2008.

VDS in the hull design, and CADDS in outfitting design. Each of the different fields of design uses distinct hardware environment and utilizes various tools for the specific design process. In addition, these systems are processed independently under distributed environment. Considering the problems above, the concept of early design is important as the early design uses concurrent design to unify different information and knowledge among different systems to generate related design information at a preliminary stage. Fig.1 shows the process of preliminary design of ships.

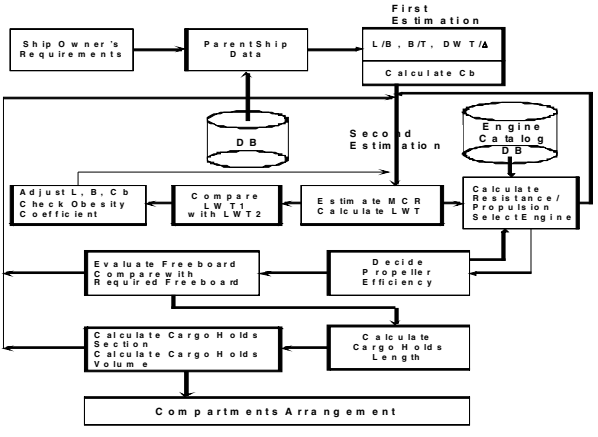


Fig. 1. The process of preliminary ship design

Many researches have been tried to solve the problem. Currently, among the different researches performed, the knowledge sharing approach which introduces the concept of agent is receiving very much of the spotlight.

In this paper, multi agent system is implemented which exchanges and shares information of the preliminary ship design process using ACL. KQML [5] is used as the outer language, Prolog is used as the inner language and CORBA, referred to as the standard distributed object environment, is used as the technique of exchanging the information in the distributed environment.

The agent-based preliminary ship design system is developed with the distributed object structure that is capable of communicating efficiently between agents using CORBA. To do this, the Caffeine function of VisiBroker is used. And the design flow control is freely achieved using the inference function of Prolog. The conversation module is added in order to use efficiently KQML message and to exchange meaningful messages. Fig.2 shows the components of the agent-based preliminary ship design system. The system implemented in this study infers using Prolog as the inner language, and implemented using the conversation module, KQML handler and CORBA.

Fig.3 shows the whole structure of the implemented multi agent system in this paper. In this system, the communication between the facilitator and each sub-agent is achieved using CORBA. Here, the facilitator trades the communication between the agents and manages the work progress. In addition, it has the inferring function and it can be referred to as a type of super agent.

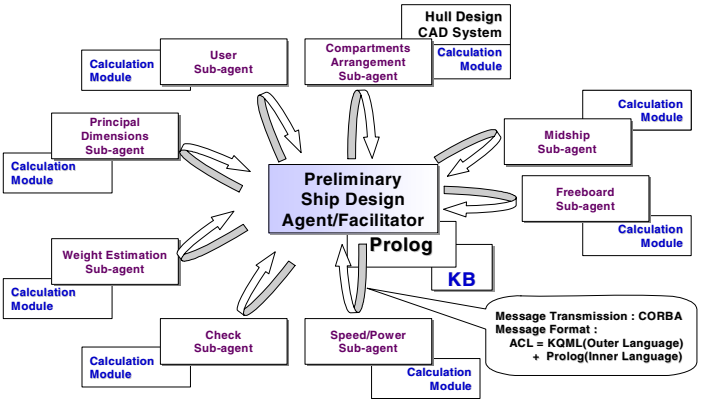


Fig. 2. Components of the implemented agent-based preliminary ship design system

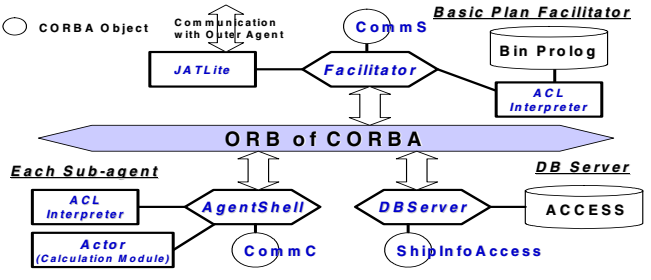


Fig. 3. System configuration of the agent-based preliminary ship design system

In this figure, the CommC and CommS provide the method for communication as a CORBA object. The ShipInfoAccess provides the method which reads and stores the real data of the ship from the database as CORBA object, as well. The connection with the database is efficiently implemented using JDBC. More efficient communication and the standard distributed environment are constructed by adopting CORBA.

3 Case-Based Conflict Resolution in Agent-Based Preliminary Ship Design System

Conflict resolution is most important in multi-agent system, and many attempts have been made. Since inefficient resolution of conflicts in decision-making can lead to high cost and low quality [6,7]. Case-Based Reasoning (CBR) approach is one of the strategies to solve the conflict problem. If there were some cases for solving similar problems previously, we can solve current conflict problems based on the previous cases without inference.

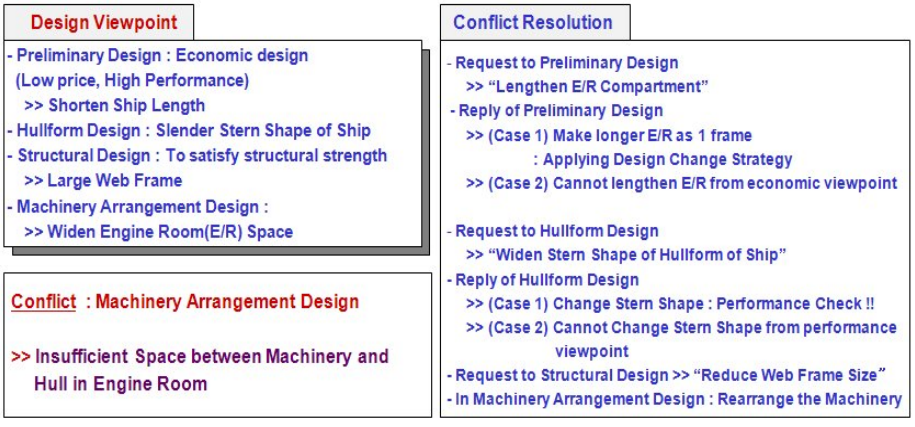


Fig. 4. A Conflict Case in Preliminary Ship Design

3.1 Typical Conflicts in Preliminary Ship Design

Owing to the different design goals of the ship design agents, conflicts can occur in the middle of cooperation of the distributed systems. In this section, the design conflicts and their resolution cases occurring in an agent-based ship design system are extracted. Although a lot of conflicts are occurring in the ship design process, only some typical conflicts are treated in this paper. Fig.4 depicts conflict problems in ship design process caused by different design notions of the related agents. That is, preliminary design agent wants to shorten the length of ship in an economic point of view. On the contrary, the hullform design agent prefers to lengthen the ship based on a performance viewpoint, such as low wave resistance, high speed, and so on. In the same manner, the machinery arrangement (MA) design agent has to arrange a great deal of machinery in the narrow space of ship engine room (E/R) located in the stern part of a ship. Hence the broader the engine room space, the better the viewpoint of layout. But the goal of the hullform design agent is a slender stern part of a ship in the viewpoint of performance. In addition, structural design agent is willing to arrange a bigger web frame in the viewpoint of strength. This makes an engine room space narrower. So many conflicts are occurring in the middle of the ship designing process caused by these different notions. In this paper, conflict resolution handler is developed so as to treat a series of the conflict process effectively [8].

3.2 Implementing Case Base and Conflict Resolution

A case is composed of three parts such as situation, solution, and the outcoming. Here, to represent the conflicts between pre-described design agents, the case is subdivided into four sets as follows.

(1) *Title*
As a symbolic meaning, a title classifies the cases according to the kinds of conflicts. The followings are described titles of implemented cases.

- Conflict between MA design agent and Preliminary Design agent
- Conflict between MA design agent and Hullform Design agent

- Conflict between MA design agent and Midship Design agent
- Conflict between MA design agent

(2) *Description*

A description is a primary factor in deciding the characteristic of case in which items for classifying the similar cases are described.

- Ship type : bulk carrier, tanker, container, etc.
- Conflict type : hull clearance, passage way, etc.
- Related agent name : Preliminary Design, MA design, etc.

(3) *Question/Answer*

After the related cases are selected from the indexing procedure based on the descriptions, question/answer is used to present the closest case through the similarity assessment for characteristic values. A weight can be given for each item.

- E/R compartment can lengthen? (Yes / No)
- Can it be possible to change hullform of stern part as full? (Yes / No)
- Can the Web frame be reduced? (Yes / No)

(4) *Action*

Action part corresponds to the solution of a given problem.

- Make E/R longer (Move FWD BHD)
  - ☞ Execute redesign process of preliminary design
  - ☞ Redesign machinery arrangement process
- Change the hullform of stern part
  - ☞ Execute evaluation process of speed/power
  - ☞ Redesign machinery arrangement process (Check hull clearance)
- Reduce web frame size
  - ☞ Execute evaluation process of structural strength

The conflicts occurred in the middle of designing process are resolved by the developed case-based conflict resolution handler [8]. This handler located in the facilitator perceives whether a conflict has occurred or not, based on the received information from agents. When a conflict has occurred, it is resolved promptly and the reply is transferred to related agents [4].

One of conflict cases and resolution by CBR approach are described as follows briefly. If a conflict is occurred in machinery design agent such like "*Insufficiency of clearance between machinery and hull of ship*", the KQML message as following is generated and sent to facilitator.

```
(tell     :sender MA_design_agent
                  :receiver facilitator
                  :language KIF
                  :reply-with MA0
                  :content (and (conflict CF1)
                                (conflict.type CF1 hullClearance)
                                (conflict.shipId CF1 h30)
                                (conflict.position CF1 (hullform.frame h30 13))
                                (conflict.source CF1 MA_design_agent)) )
```

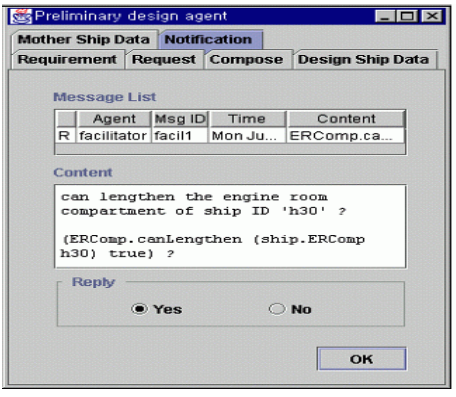


Fig. 5. Message Handling in Design Agent

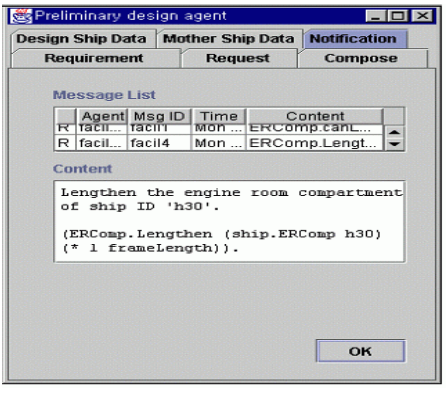


Fig. 6. Outcome of CR and its Action

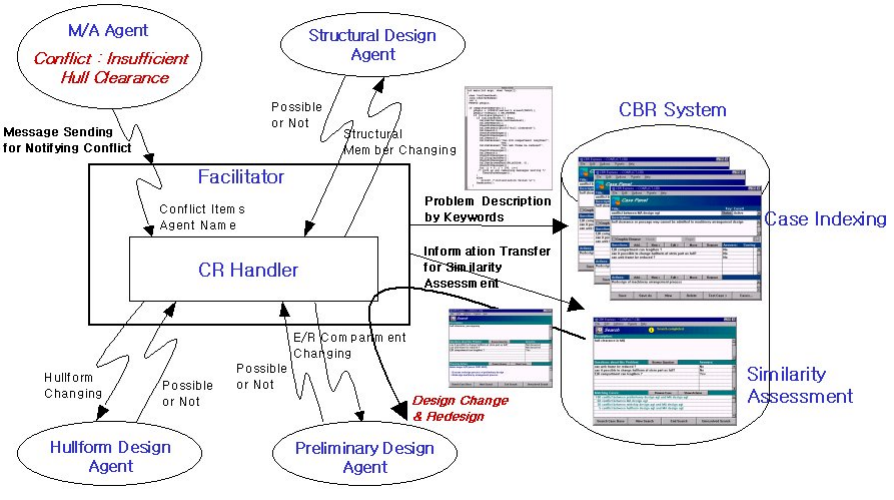


Fig. 7. Conflict Resolution Process in Agent-Based Preliminary Ship Design System

The above message is only routed from the facilitator to the handler for resolving the conflict. Problem is described for searching similar cases based on contents and agent name transferred from facilitator. This information is transferred to case-base system by using API(Application Programming Interface) functions as follows.

CallSetDescription("hull clearance in MA design agent");

In order to evaluate the similarity score of the cases, case-based system communicate with related agent by question/answer. After a while '*ask-if*' message is shown at replying window as in Fig.5 in preliminary design agent, if designer answers the question, the reply is transferred to the '*tell*' KQML message, and sent to handler via facilitator. Through these communication processes, case-based system decides the lengthening of engine room compartment as 1 frame. The message is transferred to

preliminary design agent in which re-design process is carried out as in Fig.6. Fig.7 illustrates the concept of conflict resolution in machinery design of ship and collaboration in agent-based ship design system.

4 Conclusions

In this paper, two topics are discussed in a preliminary ship design environment. One is the construction of agent-based ship design system to implement collaborative ship design environment, the second is the resolution of the conflicts that occur in the middle of cooperative ship design works. A framework of agent-based preliminary ship design system to cooperate and share the information with each other under the distributed and heterogeneous system environments is developed. And also conflict resolution strategy based on case-based reasoning technique is implemented. Case-based approach to resolve a current conflict problem based on previous similar cases is very reasonable for multi-disciplinary ship design stage.

Acknowledgement

This work is supported by Advanced Ship Engineering Research Center (R11-2002-104-08002-0).

References

1. Jha, K.N., et al.: Agent Support for Design of Aircraft Part. In: Proc. of DETC 1998 ASME Design Automation Conference, Atlanta (1998)
2. Park, H., et al.: An Agent-Based Approach to Concurrent Cable Harness Design. AIEDAM 8, 45–61 (1994)
3. Fujita, K., Akagi, S.: Agent-Based Distributed Design System Architecture for Basic Ship Design. In: Proc. of 5th ISPE International Conference on Concurrent Engineering (CE 1998), Tokyo, Japan (1998)
4. Lee, K.H., Lee, K.Y.: Case-based Conflict Resolution in Multi-Agent Ship Design System. In: Zhang, S., Jarvis, R. (eds.) AI 2005. LNCS (LNAI), vol. 3809, pp. 826–829. Springer, Heidelberg (2005)
5. Finin, T., Labrou, Y., Mayfield, J.: KQML as an agent communication language. In: Bradshaw, J.M. (ed.) Software Agents. MIT Press, Cambridge (1995)
6. Klein, M.: Supporting Conflict Resolution in Cooperative Design Systems. IEEE Trans. on Stems, Man, and Cybernetics 21(6) (1991)
7. Klein, M., Lu, S.: Conflict Resolution in Cooperative Design. Artificial Intelligence in Engineering 4(4), 168–180 (1993)
8. Lee, K.H., Lee, K.Y.: Agent-based collaborative design system and conflict resolution based on a case-based reasoning approach. In: AIEDAM, vol. 16, pp. 93–102 (2002)

Author Index