Lecture Notes in Artificial Intelligence 1805
Subseries of Lecture Notes in Computer Science

Takao Terano Huan Liu
Arbee L. P. Chen (Eds.)

Knowledge Discovery and Data Mining

Current Issues and New Applications

4th Pacific-Asia Conference, PAKDD 2000
Kyoto, Japan, April 2000
Proceedings

Springer

Lecture Notes in Artificial Intelligence 1805

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Takao Terano Huan Liu
Arbee L.P. Chen (Eds.)

Knowledge Discovery and Data Mining

Current Issues and New Applications

4th Pacific-Asia Conference, PAKDD 2000
Kyoto, Japan, April 18-20, 2000
Proceedings

Springer

Series Editors

Jaime G. Carbonell,Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Takao Terano
Universiy of Tsukuba
Graduate School of Systems Management
3-29-1 Otsuka, Bunkyo-ku,Tokyo 112-0012, Japan
E-mail: terano@gssm.otsuka.tsukuba.ac.jp

Huan Liu
Arizona State University
Department of Computer Science and Engineering
P.O. Box 875 406, Tempe, AZ, 85287-5406
E-mail: hliu@asu.edu

Arbee L.P. Chen
National Tsing Hua University
Department of Computer Science
Hsinchu, Taiwan 300, ROC
E-mail: alpchen@cs.nthu.edu.tw

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Knowledge discovery and data mining : current issues and new applications ; 4th Pacific Asia conference ; proceedings / PAKDD 2000, Kyoto, Japan, April 18 - 20, 2000. Takao Terano ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1805 : Lecture notes in artificial intelligence)
ISBN 3-540-67382-2

CR Subject Classification (1991): I.2, H.3, H.5.1, G.3, J.1

ISBN 3-540-67382-2 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag is a company in the BertelsmannSpringer publishing group.
© Springer-Verlag Berlin Heidelberg 2000
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Steingräber Satztechnik GmbH, Heidelberg
Printed on acid-free paper SPIN 10720199 06/3142 5 4 3 2 1 0

Preface

The Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2000) was held at the Keihanna-Plaza, Kyoto, Japan, April 18 – 20, 2000. PAKDD 2000 provided an international forum for researchers and application developers to share their original research results and practical development experiences. A wide range of current KDD topics were covered including machine learning, databases, statistics, knowledge acquisition, data visualization, knowledge-based systems, soft computing, and high performance computing. It followed the success of PAKDD 97 in Singapore, PAKDD 98 in Australia, and PAKDD 99 in China by bringing together participants from universities, industry, and government from all over the world to exchange problems and challenges and to disseminate the recently developed KDD techniques.

This PAKDD 2000 proceedings volume addresses both current issues and novel approaches in regards to theory, methodology, and real world application. The technical sessions were organized according to subtopics such as Data Mining Theory, Feature Selection and Transformation, Clustering, Application of Data Mining, Association Rules, Induction, Text Mining, Web and Graph Mining. Of the 116 worldwide submissions, 33 regular papers and 16 short papers were accepted for presentation at the conference and included in this volume. Each submission was critically reviewed by two to four program committee members based on their relevance, originality, quality, and clarity.

The PAKDD 2000 program was enhanced by two keynote speeches and one invited talk: Yoshiharu Sato of Hokkaido University, Japan (Statistics), Ryszard S. Michalski of George Mason University, USA (Machine Learning), and Andrew Tomkins of IBM Almaden Research Center (Databases). The PAKDD 2000 program was further complemented by five tutorials: Enterprise Data Mining with Case Studies (Zhexue Huang and Graham J Williams), Data Mining with Decision Trees (Johannes Gehrke), Knowledge Extraction from Texts - Application to Human Resources in Industry (Yves Kodratoff), Rough Sets in KDD: A Tutorial (Andrzej Skowron and Ning Zhong), and Data Mining on the World Wide Web (Wee-Keong NG). Two international workshops were co-hosted with PAKDD 2000 focusing on two KDD frontiers: the International Workshop on Web Knowledge Discovery and Data Mining (WKDDM 2000) and the International Workshop of KDD Challenge on Real-World Data (KDD Challenge 2000).

The success of PAKDD 2000 would not have been possible without the generous help rendered to us. We would like to extend our heartfelt gratitude to the program committee members and the steering committee members for their invaluable contributions. Special thanks go to the conference chairs: Hiroshi Motoda and Masaru Kitsuregawa for their leadership and involvement in making the conference run smoothly. We would like to express our immense gratitude to all the contributors to the conference for submitting and presenting papers, offering tutorials, giving talks, and organizing workshops. Special thanks are

due to Yuko Ichiki of Keihanna-Plaza Co. for her excellent secretarial work. The conference was sponsored by various academic societies in Japan. These are the Japanese Society of Artificial Intelligence (JSAI), SIG-DE (Data Engineering) and SIG-AI (Artificial Intelligence) of The Institute of Electronics, Information and Communication Engineers, SIG-DM (Data Mining) of the Japan Society for Software Science and Technology, SIG-DB (Data Base) and SIG-ICS (Intelligent and Complex Systems) of the Information Processing Society of Japan, and ACM SIG-MOD Japan. PAKDD 2000 was also generously supported by the SAS Institute, Japan and the Telecommunication Advancement Foundation (TAF), Japan.

We hope all participants had a pleasant stay at PAKDD 2000 as well as in Japan, exchanged refreshing views, and we wish them great success in their KDD endeavors.

April 2000 Takao Terano, Huan Liu, and Arbee L. P. Chen

PAKDD 2000 Conference Committee

Conference Chairs:

Masaru Kitsuregawa, University of Tokyo, Japan
Hiroshi Motoda, Osaka University, Japan

Program Chairs:

Takao Terano, Tsukuba University, Japan (Chair)
Huan Liu, Arizona State University, USA (Co-chair)
Arbee L. P. Chen, National Tsing Hua University, Taiwan (Co-chair)

Publicity Chair:

Shinichi Morishita, University of Tokyo, Japan

Workshop Chair:

Takahira Yamaguchi, Shizuoka University, Japan

Tutorial Chair:

Shusaku Tsumoto, Shimane Medical University, Japan

Local Organizing Committee Chair:

Takashi Washio, Osaka University, Japan

PAKDD Steering Committee:

Xindong Wu, Colorado School of Mines, USA (Chair)
Hongjun Lu, National University of Singapore (Co-chair)
Ramamohanarao Kotagiri, University of Melbourne, Australia
Huan Liu, Arizona State University, USA
Hiroshi Motoda, Osaka University, Japan
Lizhu Zhou, Tsinghua University, China
Ning Zhong, Yamaguchi University, Japan

PAKDD 2000 Program Committee

Akinori Abe	NTT Communication Science Laboratories, Japan
Tatsuya Akutsu	University of Tokyo, Japan
Hiroki Arimura	Kyushuu University, Japan
Ming-Syan Chen	National Taiwan University, Taiwan
David Cheung	Hong Kong University, Hong Kong
Vic Ciesielski	RMIT, Australia
Honghua Dai	Deakin University, Australia
Usama Fayyad	Microsoft Research, USA
Takeshi Fukuda	IBM, Tokyo Research Laboratory, Japan
Yike Guo	Imperial College, UK
Jiawei Han	Simon Fraser University, Canada
Tomoyuki Higuchi	Institute of Statistical Mathematics, Japan
Tu Bao Ho	Japan Advanced Institute of Science and Technolo
Howard Ho	IBM, Almaden Research Center, USA
Xiaohua Hu	Knowledge Stream, USA
Zhexue Huang	MIP, Australia
Seiji Isobe	NTT Software Corporation, Japan
Manabu Iwasaki	Seikei University, Japan
Won-Chul Jhee	Hong-Ik University, Korea
Hiroyuki Kawano	Kyoto University, Japan
Kyung-Chang Kim	Hong-Ik University, Korea
Kevin Korb	Monash University, Australia
Ramamohanarao Kotagiri	University of Melbourne, Australia
Deyi Li	Beijing System Engineering Institute, China
Jianmin Li	University of Illinois at Chicago, USA
T.Y. Lin	San Jose State University, USA
Charles X. Ling	Univ. Western Ontario, Canada
Bing Liu	National University of Singapore
Chunnian Liu	Beijing Polytechnic University, China
Jiming Liu	Hong Kong Baptist University
Hongjun Lu	National University of Singapore
Yasuhiko Morimoto	IBM, Tokyo Research Laboratory, Japan
Shinichi Morishita	University of Tokyo, Japan
Masayuki Numao	Tokyo Institute of Technology, Japan
Tadashi Ohmori	University of Electro-Communications, Japan
Yukio Ohsawa	Tsukuba University, Japan
Gregory Piatetsky-Shapiro	Knowledge Stream, USA
Mohamed Quafafou	University of Nantes, France
Zbigniew W. Ras	University of North Carolina, USA
Keun Ho Ryu	Chungbuk National University, Korea

Peter Scheuermann	Northwestern University, USA
John C. Shafer	IBM, Almaden Research Center, USA
Arun Sharma	University of New South Wales, Australia
Zhongzhi Shi	Chinese Academy of Sciences, China
Arul Siromoney	Anna University, India
Andrzej Skowron	Warsaw University, Poland
Wataru Sunayama	Osaka University, Japan
Einoshin Suzuki	Yokohama National University, Japan
Atsuhiro Takasu	NACSIS, Japan
Shiro Takata	ATR, Japan
Bhavani Thuraisingham	MITRE Corporation, USA
Kai Ming Ting	Deakin University, Australia
Hiroshi Tsukimoto	Toshiba Corporation, Japan
Shusaku Tsumoto	Shimane Medical University
Jeffrey D. Ullman	Stanford University
Lipo Wang	Nanyang Technical University, Singapore
Takashi Washio	Osaka University, Japan
Graham Williams	CSIRO, Australia
Xindong Wu	Colorado School of Mines, USA
Beat Wuthrich	Hong Kong University of Science & Technology
Takahira Yamaguchi	Shizuoka University, Japan
Yiyu Yao	University of Regina, Canada
Suk-Chung Yoon	Widener University, USA
Tetsuya Yoshida	Osaka University, Japan
Meide Zhao	University of Illinois at Chicago, USA
Zijian Zheng	Blue Martini Software, USA
Aoying Zhou	Fudan University, China
Ning Zhong	Yamaguchi University, Japan
Lizhu Zhou	Tsinghua University, China
Jan Zytkow	University of North Carolina, USA

PAKDD 2000 Reviewers

Akinori Abe
Akiko Aizawa
Tatsuya Akutsu
Hiroki Arimura
Ming-Syan Chen
David Cheung
Vic Ciesielski
Honghua Dai
Manoranjan Dash
Yi Du
Usama Fayyad
Takeshi Fukuda
Xiujun Gong
Yike Guo
Jiawei Han
Hisaaki Hatano
Lawrence. J. Henschen
Tomoyuki Higuchi
Tu Bao Ho
Howard Ho
Charles Hu
Xiaohua Hu
Zhexhe Huang
Md. Farhad Hussain
Tetsuya Iizuka
Seiji Isobe
Manabu Iwasaki
Won-Chul Jhee
Rong Jiang
Hiroyuki Kawano
Kyung-Chang Kim
Masaru Kitsuregawa
Kevin Korb
Ramamohanarao Kotagiri
Doheon Lee
Deyi Li
Jianmin Li
Xiaoli Li
T.Y. Lin
Charles X. Ling
Bing Liu
Chunnian Liu
Huan Liu
Jimin Liu
Jiming Liu
Oscar Ortega Lobo
Hongjun Lu
Yasuhiko Morimoto
Shinichi Morishita
Chie Morita
Koichi Moriyama
Hiroshi Motoda
Svetlozar Nestorov
N. Binh Nguyen
T. N. Nguyen
Masayuki Numao
Kensuke Ohmori
Tadashi Ohmori
Yukio Ohsawa
Wen Pen
Gregory Piatetsky-Shapiro
Mohamed Quafafou
Zbigniew W. Ras
Keun Ho Ryu
Makoto Sato
Peter Scheuermann
John C. Shafer
Arun Sharma
John Shepherd
Zhongzhi Shi
Hisako Shiohara
Arul Siromoney
Andrzej Skowron
Wataru Sunayama
Einoshin Suzuki
Atsuhiro Takasu
Takao Terano
Bhavani Thuraisingham
Kai Ming Ting
Hiroshi Tsukimoto
Shusaku Tsumoto
Wisut Sae-Tung
Jeffrey D. Ullman
Lipo Wang
Takashi Washio
Alicja Wieczorkowska
Graham Williams
J.Wroblewski
Xindong Wu
Beat Wuthrich
Takahira Yamaguchi
Yiyu Yao
Shiren Ye
Suk-Chung Yoon
Mariko Yoshida
Tetsuya Yoshida
Xiangtao You
Meide Zhao
Zijian Zheng
Ning Zhong
Aoying Zhou
Lizhu Zhou
Jan Zytkow

Table of Contents

Clustering

Application of Data Mining

Association Rules and Related Topics

Induction

Text, Web, and Graph Mining

Perspective on Data Mining from Statistical Viewpoints

Yoshiharu Sato

Division of Systems and Information Engineering, Hokkaido University, Sapporo, 060-8628 Japan,
ysato@main.eng.hokudai.ac.jp

Abstract. The history of statistical data analysis is old, it goes back to the 1920's. Many fundamental concepts of multivariate statistical data analysis, especially pure theoretical notions, have been accomplished by the 1950's. After the 1960's, the practical applications of multivariate statistical data analysis have been available, coupled with the progress of computers, and these have also been an affect on theoretical considerations.

The basic process of data analysis is given as follows:

p1). An objective of data analysis is given.
p2). The data which seems to be closely connected with the objective is observed. (sampling data)
p3). Constructing a model (or a set of models) for explaining the variation of the data.
p4). Preprocessing (or transforming) the original data in order to make consistency between input data and the model.
p5). Identification of the model based on observed (input) data.
p6). Evaluate a goodness of fit. If the goodness of fit is insufficient, then return to P2) or P3), else go to next process.
p7). Interpretation of the result and investigate the validity.

The most different point on "data mining" and statistical data analysis seems to be the concept of "Data". In data mining, the data is given as a database in advance. But, in statistical data analysis, the data is observed according to the objective of the analysis.

On the other hand, the object of "data mining" is to find the effective (or valuable) information in the data. From the framework of statistical data analysis above, the main processes of data mining are p3), p4) and p5). However, the concept of "efficient information" in data mining is different from the main part of the data variation in statistical data analysis. For instance, in principal component analysis, the main part of the data variation is obtained as the first principal component, which has the largest proportion. But in data mining, the major variation of the data is of no interest, because the knowledge obtained from it is trivial. Then, data mining seems to be interested in the principal components with small proportion in order to get unusual but valuable information. Hence, statistical data analysis for residual data which is removing the main part of the data variation from the original data, will be useful for data mining.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, p. 1, 2000.
© Springer-Verlag Berlin Heidelberg 2000

Inductive Databases and Knowledge Scouts

Ryszard S. Michalski

PRC Professor of Computational Sciences and Information Technoloy
Machine Learning and Inference Laboratory
Institute for Computational sciences and Informatics
George Mason University
www.mli.gmu.edu/michalski

"All human beings desire to know"
Aristotle, Metaphysics, I.1.

Abstract. The development of very large databases and the world wide web has created extraordinary opportunities for monitoring, analyzing and predicting global economical, ecological, demographic, political, and other processes in the world. Our current technologies are, however, insufficient for these tasks, and we drowning in the deluge of data that are being collected world-wide.

New methods and integrated tools are needed that can generate goal-oriented knowledge and predictive hypotheses from massive and multimedia data, stored in large distributed databases, warehouses, and the world wide web. These methods and tools must be able to cope not only with huge data volumes in various forms, but also with data inconsistency, missing values, noise, and/or possibly weak data relevance to any given task. The development of effective methods and systems for knowledge mining in large multimedia datas emerges as a central challenge on the research agenda for the 21st century.

This talk will briefly discuss a novel project toward the above goals, which is conducted in the GMU Machine Learning and Inference Laboratory. The project concerns the development of what we call inductive databases and knowledge scouts. An inductive database extends a conventional database by integrating in it inductive inference capabilities (possibly also other types of uncertain reasoning). These capabilities allow a database to answer queries that require synthesizing plausible knowledge and make hypothetical predictions.

One of the important design conditions for an inductive database is that the hypothesized knowledge satisfy the "postulate of comprehensiblity," that is, is in the form easy to understand and interpret by people. This can be achieved employing an appropriate representation langauge (for example, attributional calculus), and implementing a form of reasoning which we call "natural induction." An inductive database supports the implementation of knowledge scouts, which are personal intelligent agents that "live" in the database, and automatically search for knowledge of interest to a particular user or group of users.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 2–3, 2000.
© Springer-Verlag Berlin Heidelberg 2000

Presented concepts will be illustrated by initial results on searching for patterns that relate lifestyles with diseases in a large database from the American Cancer Society. At the end of the talk, we will demonstrate a system illustrating principles of natural induction.

Hyperlink-Aware Mining and Analysis of the Web

Andrew Tomkins

K53/B1 IBM Almaden Research Center
650 Harry Rd., San Jose, CA 95120-6099, USA
tomkins@almaden.ibm.com

Abstract. The approximately seven billion hyperlinks on the WWW, and the anchortext surronding them, represent a valuable collection of editorial information about web pages. We begin by discussing methods for incorporating this link information into web search. Next, we consider a follow-on question: is it possible to apply data mining techniques to the link structure of the web in order to discover all communities, including those that have only just formed and whose members may not yet be aware of one another.

We also consider modeling and measurement of this hyperlink structure. A recent analysis of the web graph indicates that the macroscopic structure is considerably more intricate than suggested by earlier experiments. We describe these results, and go on to discuss some progress towards defining analytical models for graphs such as the web.

The work described here is joint with Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Raymie Stata, Sridhar Rajagopalan, Eli Upfal and Janet Wiener.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, p. 4, 2000.
© Springer-Verlag Berlin Heidelberg 2000

Polynomial Time Matching Algorithms for Tree-Like Structured Patterns in Knowledge Discovery

Tetsuhiro Miyahara[1], Takayoshi Shoudai[2], Tomoyuki Uchida[1], Kenichi Takahashi[1], and Hiroaki Ueda[1]

[1] Faculty of Information Sciences,
Hiroshima City University, Hiroshima 731-3194, Japan
{miyahara@its,uchida@cs,takahasi@its,ueda@its}.hiroshima-cu.ac.jp
[2] Department of Informatics,
Kyushu University 39, Kasuga 816-8580, Japan
shoudai@i.kyushu-u.ac.jp

Abstract. Graphs have enough richness and flexibility to express discrete structures hidden in a large amount of data. Some searching methods utilizing graph algorithmic techniques have been developed in Knowledge Discovery. A term graph, which is one of expressions for graph-structured data, is a hypergraph whose hyperedges are regarded as variables. Although term graphs can represent complicated patterns found from structured data, it is hard to do pattern match and pattern search in them. We have been studying subclasses of term graphs, called regular term trees, which are suited for expressing tree-like structured data. In this paper, we consider a matching problem for a regular term tree t and a standard tree T, which decides whether or not there exists a tree T' such that T' is isomorphic to T and T' is obtained by replacing variables in t with some trees. First we show that the matching problem for a regular term tree and a tree is NP-complete even if each variable in the regular term tree contains only 4 vertices. Next we give a polynomial time algorithm for solving the matching problem for a regular term tree and a tree of bounded degree such that the regular term tree has only variables consisting the constant number of vertices greater than one. We also report some computational experiments and compare our algorithm with a naive algorithm.

1 Introduction

Graph-structured data occurs in many domains, such as biomolecular database, chemical database, the World Wide Web, or semistructured data. Many researchers try to find hidden knowledge from structures of such data by using data mining techniques. The formalization of expressing graph-structured data is quite important for finding useful knowledge [10].

A term graph, which is one of expressions of graph-structured data, is a hypergraph whose hyperedges are regarded as variables. By expressing structures

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 5–16, 2000.
© Springer-Verlag Berlin Heidelberg 2000

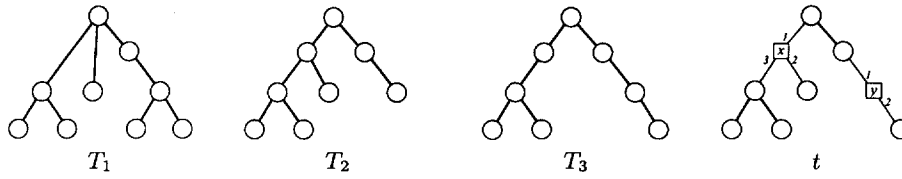


Fig. 1. A term tree t as a tree-like structured pattern which matches trees T_1, T_2 and T_3.

of data in database with term graphs, we can design tools for discovering hidden knowledge or background knowledge from graph-structured data. In Fig. 1, for example, we can obtain each tree T_1, T_2 and T_3 from the term tree t by replacing hyperedges in t with arbitrary trees. That is, the term tree t shows common structures between them. The language of first-order logic is much better suited for expressing background knowledge and a graph structure can be expressed by using first-order logic [3]. Then, inductive logic programming (ILP) systems in knowledge discovery have been proposed [1,2,4]. In [8], we designed and implemented the knowledge discovery system KD-FGS for graph-structured data, which employs Formal Graph System (FGS,[11]) as a knowledge representation language and a refutably inductive inference as an ILP mechanism [9]. FGS is a kind of logic programming system which uses term graphs instead of terms in first-order logic. Therefore FGS can directly deal with graphs and is suited for expressing background knowledge obtained from graph-structured data. By using a term graph, we can design tools based on a graph pattern matching method for finding new knowledge represented by term graphs obtained from graph-structured data. Such tools are useful for finding association rules over term graphs, producing decision trees having term graphs as vertex labels, and finding the minimum term graph by using the minimum description length principle.

In this paper, we consider a matching problem for a term graph and a graph. Informally, the matching problem for a term graph g and a graph G is to decide whether or not there exists a graph G' such that G' is isomorphic to G and G' is obtained by replacing each variable in g with an arbitrary graph. This problem is important for many knowledge discovery systems over term graphs for graph-structured data. Graphs have enough richness and flexibility to express unknown structures, but many elementary graph problems, e.g., subgraph isomorphism and largest common subgraph, are known to be NP-complete [5]. Due to this fact, it is difficult to solve the matching problem for a term graph in polynomial time. Then it is hard to design and implement a discovery system finding efficiently new knowledge from graph-structured data in practice. We consider interesting subclasses of term graphs, called regular term trees, such that their matching problems are solvable efficiently. In [7], for a regular term tree t and a tree T such that every variables in t consist of two vertices, we presented a polynomial time algorithm solving the matching problem for t and T. In this paper, we show that, in general, the matching problem for a regular term tree t and a tree T is

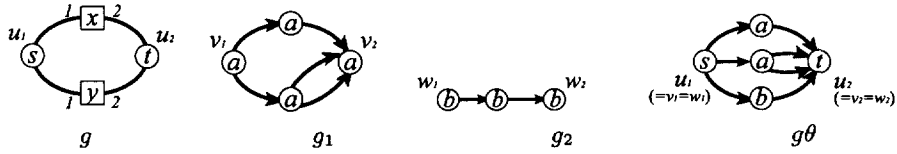


Fig. 2. A term graph $g = (V, E, H)$ is defined by $V = \{u_1, u_2\}$, $E = \emptyset$, $H = \{e_1 = (u_1, u_2), e_2 = (u_1, u_2)\}$, $\varphi_g(u_1) = s$, $\varphi_g(u_2) = t$, $\lambda_g(e_1) = x$, and $\lambda_g(e_2) = y$. $g\theta$ is obtained by applying a substitution $\theta = \{x := [g_1, (v_1, v_2)], y := [g_2, (w_1, w_2)]\}$ to g. A variable is represented by a box with lines to its elements and the order of its items is indicated by the numbers at these lines.

NP-complete even if each variable in t consists of only 4 vertices. But, if t has only variables containing constant number of vertices greater than one and T is a tree of bounded degree, we can give a polynomial time algorithm solving the matching problem. These show that a term tree is a quite useful expression of knowledge obtained from tree-like structured data.

This paper is organized as follows. In Section 2, we introduce a term graph as an expression of knowledge for graph-structural data. And a regular term tree is defined. In Section 3, we consider the matching problem for a regular term tree, and give polynomial-time algorithms solving the matching problem for some classes of regular term trees. Finally, we give a result of computational experiments comparing our algorithm presented in [7] with a naive algorithm in Section 4. Our algorithms and the computational result lead us to develop new knowledge discovery tools employing term graphs directly which express knowledge obtained from tree-like structured data.

2 Preliminaries

Let Σ and Λ be finite alphabets, and let X be an alphabet. An element in Σ, Λ and X is called a *vertex label*, *edge label* and *variable label*, respectively. Assume that $(\Sigma \cup \Lambda) \cap X = \emptyset$. A *term graph* $g = (V, E, H)$ consists of a vertex set V, an edge set E and a multi-set H. Each element in H is a list of distinct vertices in V and is called a *variable*. An item in a variable is called a *port*. And a term graph g has a vertex labeling $\varphi_g : V \to \Sigma$, an edge labeling $\psi_g : E \to \Lambda$ and a variable labeling $\lambda_g : H \to X$. For a set or a list S, the number of elements in S is denoted by $|S|$. The *dimension* of a term graph g is the maximum number of $|h|$ over all variables h in g. The *degree* of a vertex u in a term graph is the sum of the number of edges and variables containing u. A term graph $g = (V, E, H)$ is called *ground* and simply denoted by $g = (V, E)$ if $H = \emptyset$. For example, a term graph $g = (V, E, H)$ is shown in Fig. 2.

Let g be a term graph and σ a list of distinct vertices in g. We call the form $x := [g, \sigma]$ a *binding* for a variable label $x \in X$. Let $g_1, \ldots, g_n$ be term graphs. A *substitution* θ is a finite collection of bindings $\{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$, where x_i's are mutually distinct variable labels in X and each g_i has no variable

labeled with an element in $\{x_1, \ldots, x_n\}$. We obtain a new term graph f by applying a substitution $\theta = \{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$ to a term graph $g = (V, E, H)$ in the following way. For each binding $x_i := [g_i, \sigma_i] \in \theta$ $(1 \leq i \leq n)$ in parallel, we attach g_i to g by removing all variables $t_1, \ldots, t_k$ labeled with x_i from H, and by identifying the m-th vertex t_j^m of t_j and the m-th vertex σ_i^m of σ_i for each $1 \leq j \leq k$ and each $1 \leq m \leq |t_j| = |\sigma_i|$. We remark that the label of each vertex t_j^m of g is used for the resulting term graph which is denoted by $g\theta$. Namely, the label of σ_i^m is ignored in $g\theta$.

3 Matching Algorithms for Tree-Like Structured Patterns

3.1 A Regular Term Tree of Bounded Degree

A substitution $\theta = \{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$ is called a *tree substitution* if all of the g_i are trees. A term graph g is called a *term tree* if for any tree substitution θ which contains all variable labels in g, $g\theta$ is also a tree. A term tree g is called *regular* if each variable label in g occurs exactly once [6]. For example, a regular term tree $t = (\{1,2,3,4,5,6,7,8,9,10,11\}, \{\{1,2\},\{2,4\},\{2,5\},\{7,9\}\}, \{(1,3),(2,6),(2,7,8),(3,10,11)\})$ is shown in Fig. 3. In this section, we assume that a tree which is an input to our matching algorithms is an unrooted tree without a vertex label and an edge label. In the other cases, we can easily construct similar matching algorithms.

We say that T matches t if there exists a tree substitution θ such that $t\theta$ and T are isomorphic. We give polynomial-time algorithms for solving the following problem for a regular term tree of bounded dimension and a tree of bounded degree.

REGULAR TERM TREE MATCHING
Instance: A regular term tree t and a tree T.
Question: Does T match t?

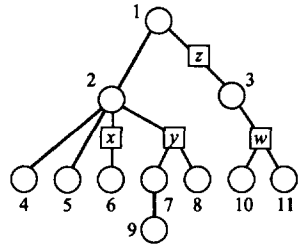
First we show the following theorem:

Theorem 1. *REGULAR TERM TREE MATCHING is NP-complete if the dimension of an input regular term tree is greater than or equal to* 4.

Proof. Membership in NP is obvious. We transform EXACT COVER BY 3-SETS (X3C) [5, page 221] to this problem.

EXACT COVER BY 3-SETS (X3C)
Instance: Set A with $|A| = 3q$ for a natural number q and a collection C of 3-element subsets of X.
Question: Does C contain an exact cover for A, i.e., a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'.

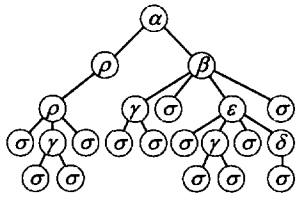
We give a transformation for a regular term tree and a tree with vertex labels. The vertex labels can be removed by replacing the vertex labels with



Term Tree t

$1 \Leftarrow 2, \{3\}$
$2 \Leftarrow 4, 5, \{6\}, \{7, 8\}$
$3 \Leftarrow \{10, 11\}$
$7 \Leftarrow 9$

Labeling Rules R_1



Tree T

$\sigma = \{4, 5, 6, 8, 9, 10, 11\}$
$\delta = \{7, \{6\}, \{8\}, \{10, 11\}\}$
$\gamma = \{3, \{6\}, \{7, 8\}, \{10, 11\}\}$
$\rho = \{\{3\}, \{6\}, \{7, 8\}, \{10, 11\}\}$
$\varepsilon = \{2, \{3\}, \{6\}, \{7, 8\}, \{10, 11\}\}$
$\beta = \{1, 2, \{3\}, \{6\}, \{7, 8\}, \{10, 11\}\}$
$\alpha = \{1, \{3\}, \{6\}, \{7, 8\}, \{10, 11\}\}$

The resulting labels by R_1

Fig. 3. An example: the labeling rule constructed from a term tree t and the resulting labels of a tree T after the procedure Matching terminates.

special trees each of which corresponds to each vertex label, for example, binary trees with a linear chain of bounded length.

Let $A = \{a_1, \ldots, a_n\}$ where $n = 3q$. Let $C = \{c_1, \ldots, c_m\}$ where $c_i \subseteq A$ with $|c_i| = 3$ and let $c_i = \{c_{i1}, c_{i2}, c_{i3}\}$ for $i = 1, \ldots, m$. The corresponding instance of REGULAR TERM TREE MATCHING is constructed in the following way. Let t be a regular term tree (V_t, E_t, H_t) where $V_t = \{v\} \cup \bigcup_{i=1}^{m} \{v_{i1}, v_{i2}, v_{i3}\}$, $E_t = \emptyset$ and $H_t = \bigcup_{i=1}^{m} \{h_i = (v, v_{i1}, v_{i2}, v_{i3})\}$. Let $\Sigma = \{a, b\} \cup \{a_1, \ldots, a_n\}$ where a and b are special vertex labels which do not appear in $\{a_1, \ldots, a_n\}$. The vertex labeling $\varphi_t : V_t \to \Sigma$ is defined as $\varphi_t(v) = a$ and $\varphi_t(v_{ij}) = c_{ij}$ for $i = 1, \ldots, m$ and $j = 1, 2, 3$. The variable labeling $\lambda_t : H_t \to X = \{x_1, \ldots, x_m\}$ is defined as $\lambda_t(h_i) = x_i$ for $i = 1, \ldots, m$. Let T be a tree (V_T, E_T), where

$V_T = \{u\} \cup \{u_1, \ldots, u_n\} \cup \{w_1, \ldots, w_{m-q}\} \cup \bigcup_{i=1}^{m-q} \{w_{i1}, \ldots, w_{in}\}$, and

$E_T = \{\{u, u_1\}, \ldots, \{u, u_n\}\} \cup \{\{u, w_1\}, \ldots, \{u, w_{m-q}\}\}$
$\cup \bigcup_{i=1}^{m-q} \{\{w_i, w_{i1}\}, \ldots, \{w_i, w_{in}\}\}$.

The vertex labeling $\varphi_T : V_T \to \Sigma$ is defined as $\varphi_T(u) = a$, $\varphi_T(w_i) = b$ for $i = 1, \ldots, m-q$, and $\varphi_T(u_j) = \varphi_T(w_{ij}) = a_j$ for $i = 1, \ldots, m-q$ and $j = 1, \ldots, n$.

Let $T_1 = (\{s_0, s_1, s_2, s_3\}, \{\{s_0, s_1\}, \{s_0, s_2\}, \{s_0, s_3\}\})$ with no label and $T_2 = (\{s_0, s_1, \ldots, s_n, s_{n+1}\}, \{\{s_0, s_{n+1}\}, \{s_1, s_{n+1}\}, \ldots, \{s_n, s_{n+1}\}\})$ with no label. We assume that there is a subcollection $C' \subseteq C$ such that every element of A occurs in exactly one member of C'. Let $\theta_1 = \{x_i := [T_{i1}, (s_0, s_1, s_2, s_3)] \mid c_i \in C'\}$ where T_{i1} is a tree T_1 with a vertex labeling φ defined as $\varphi(s_0) = a$ and

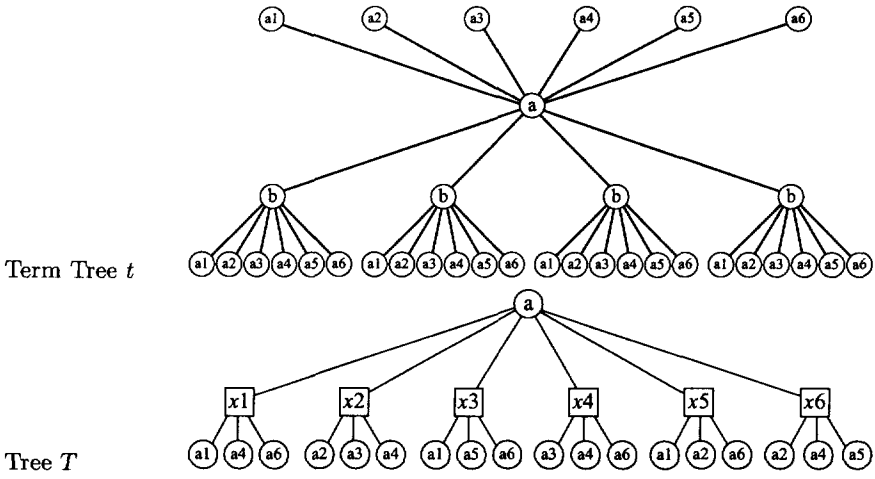


Fig. 4. A transformation from an instance (A, C) of X3C to an instance (t, T) of REGULAR TERM TREE MATCHING. $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}, C = \{c_1, c_2, c_3, c_4, c_5, c_6\}, c_1 = \{a_1, a_4, a_6\}, c_2 = \{a_2, a_3, a_4\}, c_3 = \{a_1, a_5, a_6\}, c_4 = \{a_3, a_4, a_6\}, c_5 = \{a_1, a_2, a_6\}, c_6 = \{a_2, a_4, a_5\}$.

$\varphi(s_j) = c_{ij}$ for $j = 1, 2, 3$. Let $\theta_2 = \{x_i := [T_{i2}, (s_0, s_{\delta(i,1)}, s_{\delta(i,2)}, s_{\delta(i,3)})] \mid c_i = \{c_{i1}, c_{i2}, c_{i3}\} \in C - C'$ and $c_{ij} = a_{\delta(i,j)}$ for $j = 1, 2, 3\}$, where T_{i2} is a tree T_2 with a vertex labeling φ defined as $\varphi(s_0) = a$, $\varphi(s_{n+1}) = b$ and $\varphi(s_i) = a_i$ for $i = 1, \ldots, n$, and δ is a function with $c_{ij} = a_{\delta(i,j)}$. Then $\theta = \theta_1 \cup \theta_2$ is a tree substitution such that T and $t\theta$ are isomorphic.

Conversely we assume that there is a tree substitution θ such that $t\theta$ and T become isomorphic. All $3m+1$ vertices in t have to match vertices in T. For each $i = 1, \ldots, m - q$, at most 3 vertices in $\{w_{i1}, \ldots, w_{in}\}$ can match vertices in t. Therefore at most $n+3(m-q)+1 = 3m+1$ vertices in T can match vertices in t. The bindings in θ are divided into two kinds of bindings $x_i := [T_1, (s_0, s_1, s_2, s_3)]$ and $x_i := [T_2, (s_0, s_{i_1}, s_{i_2}, s_{i_3})]$, where $1 \leq i_1, i_2, i_3 \leq n$ because the other kind of bindings can not achieve the needed number of vertex matchings $3m + 1$. Let $C' = \{c_i \mid$ there is a binding $x_i := [T_1$ with a vertex labeling, $(s_0, s_1, s_2, s_3)]$ in $\theta\}$. Then every element of A occurs in exactly one member of C'. □

Second we explain the algorithm Matching (Fig. 5) which is a framework for deciding whether a tree T matches a regular term tree t.

Let $t = (V_t, E_t, H_t)$ and $T = (V_T, E_T)$ be a regular term tree and a tree, respectively. We distinguish one vertex r_t of a term tree t and call that vertex the *root* of t. A vertex of degree one is called a *leaf* if it is not the root. A *path* from v_1 to v_i is a sequence $v_1, v_2, \ldots, v_i$ of distinct vertices such that for $1 \leq j < i$, there exists an edge or a variable which includes v_j and v_{j+1}. If there is an edge or a variable which includes v and v' such that v' lies on the path from the root r_t to v, then v' is said to be the *father* of v and v is a *child* of v'.

procedure Matching(regular term tree t, tree T);
begin
  Let r_t be one of vertices in t, which is called the root of t;
  Construct the set of all labeling rules R_{r_t};
  foreach vertex r of T, which is called the root of T **do begin**
    Label each leaf of T with the set of all leaves of t;
    while there exists a vertex v in T
        such that v is not labeled and all children of v are labeled
      do Labeling(v, R_{r_t});
    if the label of r includes r_t **then** T matches t **and exit**
  end;
  T does not match t
end.

Fig. 5. A framework for deciding whether a tree T matches a regular term tree t.

In particular for a variable h, v is said to be a *child port* of h if there is a vertex v' such that both v and v' belong to h and v is a child of v'. A *descendant* of v is any vertex on the path from v to one of the leaves of the tree.

In Matching (Fig.5), a *label* for a vertex in T is a set $\{v_1, \ldots, v_k, V_1, \ldots, V_\ell\}$ where $k \geq 0$, $\ell \geq 0$, v_i is a vertex in t, and V_j is a set of vertices in t. Let $L_1, \ldots, L_m$ be a collection of labels. For any $V' \subseteq V_t$, we say that $L_1, \ldots, L_m$ *covers* V' if there exist distinct indices $k_1, \ldots, k_{m'}, \ell_1, \ldots, \ell_{m''}$ among $1, \ldots, m$ and also there exist $v'_i \in L_{k_i}$ and $V''_j \in L_{\ell_j}$ for each $1 \leq i \leq m'$ and $1 \leq j \leq m''$ such that $V' \subseteq \{v'_1, \ldots, v'_{m'}\} \cup V''_1 \cup \cdots \cup V''_{m''}$. In particular if there is no proper subcollection of $L_1, \ldots, L_m$ which covers V' then we say that $L_1, \ldots, L_m$ *exactly covers* V'.

Let W be a set of vertices in t. The *induced term tree* of t by W is a term tree $t[W] = (W', E_t[W'], H_t[W'])$ where $W' = \{v \in V_t \mid v$ is in W or there is a vertex v' in W such that v is a descendant of $v'.\}$, $E_t[W'] = \{\{u, v\} \in E_t \mid u \in W'$ and $v \in W'\}$ and $H_t[W'] = \{(v_1, \ldots, v_n) \mid v_i \in W'$ and $(v_1, \ldots, v_n)$ is the maximal sublist of some $h \in H_t$ with keeping the order of items in $h.\}$. For a single vertex $w \in V_t$, the induced term tree $t[w]$ of t by w is defined as $t[\{w\}]$. A *corresponding induced term tree of t to $u \in V_T$* is an induced term tree by $W \subseteq V_t$ or $w \in V_t$ which matches $T[u]$, i.e., the subtree of T with the root u. In particular if the induced term tree is induced by a single vertex w, the matching between $t[w]$ and $T[u]$ has a correspondence of w to u.

Let r_t be the root of t. First we construct the set of all labeling rules.

[Basic Labeling Rules]
Let v be a vertex in t which is not a leaf. Let $v_1, v_2, \ldots, v_k$ be all children of v which are connected to v with edges. Let $h_1, h_2, \ldots, h_\ell$ be all variables which include v, and for $i = 1, \ldots, \ell$, V_i be the set of all children of v which are connected to v with the variable h_i. The labeling rule for v is defined as follows. If there is no variable which includes v, then let the generating rule of v be $v \leftarrow v_1, \ldots, v_k$, otherwise $v \Leftarrow v_1, \ldots, v_k, V_1, \ldots, V_\ell$.

procedure Labeling(vertex $u \in V_T$, set of labeling rules R_{r_t});
begin
  $L := \emptyset$;
  Let m be the number of children of u and $L_1, \ldots, L_m$ be the labels of the children;
  /* Step 1 */
  foreach $v \leftarrow v_1, \ldots, v_m$ in R_{r_t} **do**
    if $L_1, \ldots, L_m$ exactly covers $\{v_1, \ldots, v_m\}$ **then** $L := L \cup \{v\}$;
  /* Step 2 */
  foreach $v \Leftarrow v_1, \ldots, v_k, V_1, \ldots, V_\ell$ in R_{r_t} **do**
    if $L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup V_1 \cup \cdots \cup V_\ell$ **then** $L := L \cup \{v\}$;
  /* Step 3 */
  foreach variable h in t **do begin**
    Let V' be the set of all child ports of h;
    foreach $V'' \subseteq V'$ with $V'' \neq \emptyset$ **do begin**
      foreach $v \in V' - V''$ **do**
        if v and V'' satisfy either
          (1) v is a leaf and V'' is a maximal subset such that $L_1, \ldots, L_m$ covers V'', or
          (2) v is the head of a rule $v \Leftarrow v_1, \ldots, v_k, V_1, \ldots, V_\ell$ in R_{r_t} and V'' is a maximal subset such that $L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup V_1 \cup \cdots \cup V_\ell \cup V''$.
        then $L := L \cup \{V'' \cup \{v\}\}$;
      if there is no vertex v which satisfies either (1) or (2) and V'' is a maximal subset which is covered by $L_1, \ldots, L_m$
      then $L := L \cup \{V''\}$
    end
  end;
  Attach L to u as the label
end;

Fig. 6. Labeling: a procedure for labeling a vertex in T with a set of vertices in t.

Then we obtain the following theorem.

Theorem 2. *For a regular term tree t of dimension p and a tree T of degree d, REGULAR TERM TREE MATCHING is solvable in $O(N^2 n 2^p \mathrm{TMIS}(d^{O(p)}))$ time where n and N are the numbers of vertices in t and T respectively, and $\mathrm{TMIS}(s)$ is the time needed to find the maximum independent set in a graph of size s.*

Proof. In order to show the correctness of the Matching algorithm, it suffices to show the following claim.

Claim: For any $u \in V_T$, $\{t[W] \mid W \in L(u)\}$ is equal to the set of all corresponding induced term tree of t to u, where $L(u)$ is the output of Labeling procedure for u. The claim is shown by induction on the way of tree labeling of T with the root r. Suppose that u is a leaf of T. The claim holds for u, due to the labeling of a leaf in Matching. Suppose that u is not a leaf of T. Let $T[W]$ be a corresponding induced term tree of t to u. Note that (i) W is a vertex in

Input: labels $L_1, \ldots, L_m$, vertices $v_1, \ldots, v_k$, and sets of vertices $V_1, \ldots, V_\ell$;
Note that each V_i $(i = 1, \ldots, \ell)$ is the set of all child ports of a certain variable and each label L_j $(j = 1, \ldots, m)$ contains at most one subset of V_i. If the input term tree t is of bounded dimension and the input tree T is of bounded degree, the size of the graph $\mathcal{G}$ constructed below is bounded because both k and ℓ are bounded by some constants and the size of each L_i is also bounded.
begin
Construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in the following way:
$\mathcal{V}_i := \{(v_i, \{j\}) \mid v_i \in L_j \ (1 \leq j \leq m)\}$,
Let $\mathcal{E}_i$ be the complete graph constructed by $\mathcal{V}_i$,
$\mathcal{V}'_i := \{(V_i, \{j_1, \ldots, j_{m'}\}) \mid \{j_1, \ldots, j_{m'}\} \subseteq \{1, \ldots, m\}$ and $L_{j_1}, \ldots, L_{j_{m'}}$ covers $V_i\}$,
Let $\mathcal{E}'_i$ be the complete graph constructed by $\mathcal{V}'_i$,
$\mathcal{V} := \bigcup_{i=1}^{k} \mathcal{V}_i \cup \bigcup_{i=1}^{\ell} \mathcal{V}'_i$,
$\mathcal{E} := \{\{(X, Y), (X', Y')\} \mid (X, Y), (X', Y') \in \mathcal{V},\ X \neq X'$ and $Y \cap Y' \neq \emptyset\}$
$\cup \bigcup_{i=1}^{k} \mathcal{E}_i \cup \bigcup_{i=1}^{\ell} \mathcal{E}'_i$.
if there is an independent set of size $k + \ell$ for the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ **then**
$L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup V_1 \cup \cdots \cup V_\ell$
end;

Fig. 7. A procedure for determining whether or not $L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup V_1 \cup \cdots \cup V_\ell$ (Step 2 and Step 3 (2)).

t or (ii) W is a non-empty subset of the set of all child ports of a variable in t. By induction hypothesis, the claim holds for any child u' of u. In case (i) W is included in $L(u)$ due to Step 1 or 2 in Labeling. In case (ii) W is included in $L(u)$ due to Step 3 in Labeling. Then the claim holds for u.

Since the degree of t have to be less than or equal to d, the maximum length of the body of any labeling rule is less than or equal to d. Since the number of the children of any vertex is less than d and the size of each set which appears in the algorithm is less than p, the constructed graph in the procedure (Fig.7) is of size $O(d^{O(p)})$. Then the procedure (Fig.7) runs in $O(\mathrm{TMIS}(d^{O(p)}))$ time. The numbers of calls to the procedure Labeling in Matching (Fig.5) is $O(N^2)$. The numbers of calls to the procedure (Fig.7) in Labeling is $O(n2^p)$. Then the procedure Matching solves REGULAR TERM TREE MATCHING in $O(N^2 n 2^p \mathrm{TMIS}(d^{O(p)}))$ time. □

Corollary 1. *For a regular term tree t of bounded dimension and a tree T of bounded degree, REGULAR TERM TREE MATCHING is solvable in $O(N^2 n)$ time where n and N are the numbers of vertices in t and T respectively.*

If the dimension of t is equal to 2, the procedure Matching by using the procedure (Fig.8) instead of the procedure (Fig.7) solves REGULAR TERM TREE MATCHING in polynomial time.

Input: labels $L_1, \ldots, L_m$, vertices $v_1, \ldots, v_k$, and sets of vertices $\{v'_1\}, \ldots, \{v'_\ell\}$;
begin
Construct a bipartite graph $\mathcal{B} = (\mathcal{V}, \mathcal{V}', \mathcal{E})$ in the following way:
$\mathcal{V} := \{v_1, \ldots, v_k, v'_1, \ldots, v'_\ell\}$, $\mathcal{V}' := \{1, \ldots, m\}$,
$\mathcal{E}_i := \{\{v_i, j\} \mid v_i \in L_j \ \ (1 \leq j \leq m)\}$ $\quad (i = 1, \ldots, k)$,
$\mathcal{E}'_i := \{\{v'_i, j\} \mid v'_i \in L_j$ or $\{v'_i\} \in L_j \ \ (1 \leq j \leq m)\}$ $\quad (i = 1, \ldots, \ell)$,
$\mathcal{E} := \bigcup_{i=1}^{k} \mathcal{E}_i \cup \bigcup_{i=1}^{\ell} \mathcal{E}'_i$.
if for the bipartite graph $(\mathcal{V}, \mathcal{V}', \mathcal{E})$, there exists a graph matching which contains all vertices in $\mathcal{V}$
then $L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup \{v'_1\} \cup \cdots \cup \{v'_\ell\}$
end;

Fig. 8. A procedure for determining whether or not $L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup \{v'_1\} \cup \cdots \cup \{v'_\ell\}$ (for an input regular term tree whose dimension is 2).

Theorem 3 (Miyahara, et. al [7]). *If the dimension of a regular term tree is equal to 2, there exists a polynomial-time algorithm for solving REGULAR TERM TREE MATCHING.*

Since the maximum matching for a bipartite graph $\mathcal{B} = (\mathcal{V}, \mathcal{V}', \mathcal{E})$ is found in $O(|\mathcal{E}|\sqrt{\max\{|\mathcal{V}|, |\mathcal{V}'|\}})$ time, the procedure (Fig.8) runs in $O(nN^{1.5})$ time. The total time complexity of Matching is $O(n^2N^{3.5})$ if the dimension of t is equal to 2. There is a gap between the dimensions 2 and 4. The time complexity of REGULAR TERM TREE PROBLEM is still open if the dimension is 3.

3.2 Variants of Regular Term Tree

Ordered Term Tree

A rooted tree is said to be an *ordered tree* if for each vertex in the tree, the children of the vertex are ordered. In a similar way, we can construct a model of ordered term trees by giving an order to the children of each inner vertex. Ordered trees are often used to express discrete structures in the fields of natural science. For example, it is well known that RNA sequences can be expressed with labeled ordered trees. Now we are designing a knowledge discovery system for ordered tree-like structured data.

Regular Term Tree Graph with Property Π

Let Π be a property on graph G. Examples of such properties include "G is a tree", "G is planar", and "G is outer planar." A term graph g is called a *term tree graph with property* Π if for any *tree substitution* θ which contains all variable labels in g, $g\theta$ is also a graph which has property Π. The matching problem is closely related to graph isomorphism problem. For certain special subclasses of graphs, the isomorphism problem is efficiently solvable, for example, whether two planar graphs are isomorphic or not is solvable in polynomial time. Our objects in knowledge discovery are to find rich properties Π with which the matching problems for regular term tree graph are solvable efficiently.

Table 1. Experimental results of the two matching algorithms.

No.	1	2	3	4	5	6
Tree (#vertex)	g1(6)	g2(7)	g3(8)	g4(9)	g5(11)	g6(17)
Term Tree (#vertex,#variable)	t4(6,3)	t4(6,3)	t4(6,3)	t4(6,3)	t4(6,3)	t4(6,3)
Matching	true	true	true	true	true	true
Our Algorithm (Run Time, secs)	0.030	0.040	0.130	0.060	0.070	0.180
Naive Algorithm (Run Time, secs)	2.100	49.380	980.440	(aborted)	(aborted)	(aborted)

No.	7	8	9	10	11	12
Tree (#vertex)	g3(8)	g3(8)	g3(8)	g3(8)	g3(8)	g3(8)
Term Tree (#vertex,#variable)	t1(6,1)	t2(6,1)	t3(6,2)	t4(6,3)	t5(6,3)	t6(6,4)
Matching	false	false	true	true	true	true
Our Algorithm (Run Time, secs)	0.130	0.130	0.050	0.130	0.060	0.030
Naive Algorithm (Run Time, secs)	87.580	308.130	234.040	980.440	(aborted)	(aborted)

4 Implementation and Experimental Results

In order to show that our matching algorithm is useful for knowledge discovery from tree-like structured data, we have implemented our matching algorithm for a regular term tree with two ports and a tree. We have experiments of running our matching algorithm and a naive matching algorithm for such a term tree and a tree, and have compared the performance of the two algorithms.

In Table 1, we summarize some experiments of running the two algorithms on a SUN workstation Ultra-10 with clock 333 MHz. For example, in Exp. 3, the two algorithms are given as inputs a tree $g3$ with 8 vertices and a term tree $t4$ with 6 vertices and 3 variables. The value "true" in the Matching field means that the tree $g3$ is matched with the term tree $t4$. Our algorithm runs in 0.130 secs (run time) and returns "true". The naive algorithm runs in 980.440 secs (run time) and returns "true". The value "false" in the Matching field in Exp.7 and 8 means that the tree is not matched with the term tree. In Exp. 4,5,6,11 and 12, the execution of the naive algorithm is aborted without a return value after a long time of execution.

In Exp. from 1 to 6, an input term tree is fixed and an input tree is varied. These experiments show that the execution time of our algorithm slightly increases when the size of an input tree become large, but that of the naive algorithm sharply increases. In Exp. from 7 to 12, an input tree is fixed and an input term tree is varied. These experiments show that the execution time of our algorithm slightly increases when the number of variables in an input term tree become large, but that of the naive algorithm sharply increases.

All these experiments show that our matching algorithm is efficient and useful in discovering tree-like structured patterns. Our matching algorithm can be incorporated not only in the KD-FGS system but also in other knowledge discovery systems from tree-like structured data which have other knowledge representation methods such as association rule, decision diagram and so on.

5 Conclusions

We have given an algorithmic foundation of discovering knowledge from tree-like structured data. We have presented polynomial time matching algorithms for tree-like structured patterns. Computational experiments of comparing our matching algorithm and a naive matching algorithm have shown that our matching algorithm is efficient and useful. We will incorporate the matching algorithm in the KD-FGS system and other knowledge discovery systems from tree-like structured data.

Acknowledgments

This work is partly supported by Grant-in-Aid for Scientific Research (11780279) from the Ministry of Education, Science, Sports and Culture, Japan and Grant for Special Academic Research (9963,9983) from Hiroshima City University.

References

1. S. Džeroski. Inductive logic programming and knowledge discovery in databases. *Advances in Knowledges Discovery and Data Mining, MIT Press*, pages 118–152, 1996.
2. S. Džeroski, N. Jacobs, M. Molina, C. Moure, S. Muggleton, and W. V. Laer. Detecting traffic problems with ILP. *Proc. ILP-98, Springer-Verlag, LNAI 1446*, pages 281–290, 1998.
3. L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3:7–36, 1999.
4. L. Dehaspe, H. Toivonen, and R. King. Finding frequent substructures in chemical compounds. *Proceedings of the Third International Conference Knowledge Discovery and Data Mining, AAAI Press*, pages 30–36, 1998.
5. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.
6. S. Matsumoto, Y. Hayashi, and T. Shoudai. Polynomial time inductive inference of regular term tree languages from positive data. *Proc. ALT-97, Springer-Verlag, LNAI 1316*, pages 212–227, 1997.
7. T. Miyahara, T. Shoudai, T. Uchida, T. Kuboyama, K. Takahashi, and H. Ueda. Discovering new knowledge from graph data using inductive logic programming. *Proc. ILP-99, Springer-Verlag, LNAI 1634*, pages 222–233, 1999.
8. T. Miyahara, T. Uchida, T. Kuboyama, T. Yamamoto, K. Takahashi, and H. Ueda. KD-FGS: a knowledge discovery system from graph data using formal graph system. *Proc. PAKDD-99, Springer-Verlag, LNAI 1574*, pages 438–442, 1999.
9. Y. Mukouchi and S. Arikawa. Towards a mathematical theory of machine discovery from facts. *Theoretical Computer Science*, 137:53–84, 1995.
10. H. Toivonen. On knowledge discovery in graph-structured data. *Proceedings of the PAKDD Workshop on Knowledge Discovery from Advanced Databases (KDAD-99)*, 1999.
11. T. Uchida, T. Shoudai, and S. Miyano. Parallel algorithm for refutation tree problem on formal graph systems. *IEICE Transactions on Information and Systems*, E78-D(2):99–112, 1995.

Fast Discovery of Interesting Rules

Nobuhiro Yugami, Yuiko Ohta, and Seishi Okamoto

Fujitsu Laboratories
2-2-1 Momochihama, Sawaraku, Fukuoka, 814-8588, Japan
yugami@flab.fujitsu.co.jp

Abstract. Extracting interesting rules from databases is an important field of knowledge discovery. Typically, enormous number of rules are embedded in a database and one of the essential abilities of discovery systems is to evaluate interestingness of rules to filter out less interesting rules. This paper proposes a new criterion of rule's interestingness based on its exceptionality. This criterion evaluates exceptionality of rules by comparing their accuracy with those of simpler and more general rules. We also propose a disovery algorithm, DIG, to extract interesting rules with respect to the criterion effectively.

1 Introduction

The purpose of knowledge discovery system is to discover interesting patterns in a given database. There exist many types of patterns and this paper focuses on discovery of classification rules from a set of training instances represented by attribute values and class labels. A classification rule restricts values of attributes in its body and predicts a class of an instance that satisfies the body. Typically, the number of classification rules embedded in the given database is quite large and one of the essential abilities of the knowledge discovery system is to extract only interesting rules and to filter out uninteresting ones.

One approach to filtering rules is to constrain patterns of rules explicitly. Srikant et al.[11] applied this approach to association rules mining in which an user restricts what kind of items can appear in rules' bodies and heads. This approach is practical in many applications but tends to discover only rules that the user expects beforehand. In addition, even if the constraints on rules/items are given, a discovery system may output too many rules to be checked by the user. Another approach is to evaluate interestingness of rules with pre-defined criteria based on statiscal charactaristics in the target database and to accept rules with high scores with respect to the criteria. This paper belongs to this second approach.

The interestingness of rules strongly depends on what a user of the system already knows and what he/she wants to do with the discovered rules. If the user has no background knowledge of the domain represented by the target database, then he/she will prefer general rules that summarize the characteristics of each class and classify many training instances accurately[5,6]. In such a case, entropy based criteria such as information gain[4,7] and J-measure[9] can evaluate rules

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 17–28, 2000.
© Springer-Verlag Berlin Heidelberg 2000

well. However, such general rules are not appropriate when domain experts use the system because they usually know the general rules.

Following is an example from mushroom database in UCI Repository[2]. This database includes about 8,000 instances and each instance is classified into *edible* or *poisonous*. This database is very easy as a benchmark of supervised learning algorithms because the following two rules can classify most instances correctly.

$$odor \in \{almond, anise, none\} \rightarrow edible : 98\%,$$
$$odor \in \{creosote, fishy, foul, musty, pungent, spicy\} \rightarrow poisonous : 100\%,$$

where ratios after ":"are accuracy of the rules. Entropy based criteria of rules give high scores for these rules because each rule covers about a half of instances in the database and achieves very high accuracy. These rules are useful for users who don't know anything about mushrooms but are obvious for the domain experts. The purpose of this paper is *not* to learn such trivial rules but to discover unexpected, exceptional rules such as

$$cap\_color \in \{brown, red\} \wedge stalk\_root = bulbous \rightarrow edible : 100\%,$$

where

$$cap\_color \in \{brown, red\} \rightarrow edible : 50\%,$$
$$stalk\_root = bulbous \rightarrow edible : 51\%.$$

Because 52% of instances in the database are *edible*, each of two conditions, $cap\_color \in \{brown, red\}$ and $stalk\_root = bulbous$, is not related to *edible* by itself but the conjunction of them concludes *edible* with probability 100%. This type of rule is not a straightforward conclusion from correlations between attributes and classes. It represents exceptions in the given database and may be interesting for human experts. However, the traditional criteria of rules such as information gain give very low score to this type of exceptional rules because they cover only a small number of instances.

To extract exceptional rules, we need new criterion of rules. Silberschatz and Tuzhilin[8] discussed unexpectedness of rules and defined it by how the rule contradicts user's knowledge. A problem of their approach is that a discovery system has to know what its user knows. One solution to this problem is to evaluate unexpectedness by how the rule contradicts other rules held in a given database instead of user's knowledge.

Suzuki [10] proposed a discovery system PEDRE that tries to discover pairs of general rules and their exceptions. An exceptional rule is a specialization of a general rule but concludes a different class with high accuracy. General rules cover relatively many instances and may be trivial for human experts, but exceptional rules may give new knowledge to the experts. PEDRE evaluates an exceptional rule by comparing with only one general rule, its pair. There may be a different generalization of the excetional rule that predicts the same class with the exceptional rule. In such a case, the conclusion (class) of the exceptional rule holds in a large region of an instance space and the rule represents only a part

of the region. Such rule is too specific to predict its concluding class and is not appropriate as an exceptional rule.

In this paper, we propose a new criterion of interestingness to identify a rule corresponding to an isolated exceptional region by comparing its accuracy with plural general rules. In addition, our criterion can evaluate a rule that permits plural values for each attribute appearing in its body. We also propose a discovery algorithm, DIG(Discover Interesting rules with Grouping attribute values), to extract interesting rules w.r.t. the criterion effectively.

2 Interestingness of Rules

This paper focuses on discovery of classification rules from a set of labeled instances represented by attribute values. To simplify discussion, we assume all attributes are nominal and there is no missing attribute value. In this section, we first explain classification rules we deal with and then discuss how to evaluate their interestingness.

2.1 Classification Rules

A classification rule is a if-then rule whose head (conclusion) is a class label and whose body is a conjunction of conditions of attribute values. We deal with a following type of classification rules.

$$R1 : a_{i_1} \in D_{i_1} \wedge \cdots \wedge a_{i_L} \in D_{i_L} \rightarrow c,$$

where c is a certain class and D_{i_k} is a subset of possible values of attribute a_{i_k}. Many discovery systems such as ITRULE[9] and PEDRE[10] only extract rules that permit or prohibit one value for each attribute in the body of the rule, but we extract rules which allow plural values for each attribute. Grouping attribute values and allowing all values in one group increases the number of possible rules and may degrade efficiency of discovery systems. However, it is quite useful to improve readability of extracted rules because one rule with value grouping represents plural rules without grouping.

In the following discussion, we will use support and accuracy of classification rules. Support is a probability that both of a body and a head are satisfied, $P(a_{i_1} \in D_{i_1} \wedge \cdots \wedge a_{i_L} \in D_{i_L} \wedge c)$. Accuracy is a conditional probability that a head is satisfied on the condition that a body is satisfied, $P(c|a_{i_1} \in D_{i_1} \wedge \cdots \wedge a_{i_L} \in D_{i_L})$.

2.2 Interestingness without Grouping Attribute Values

We first discuss interestingness of rules without grouping attribute values, i.e. the rules in which D_{i_k} involves exactly one value v_{i_k} for each k. Then a rule without grouping is

$$R2 : a_{i_1} = v_{i_1} \wedge \cdots \wedge a_{i_L} = v_{i_L} \rightarrow c.$$

For simplicity, we use "primitive rules" to stand for the rules without grouping attribute values. Our basic idea of interestingness is that a rule is interesting if its accuracy is higher than predicted from more general rules. The larger the difference is, the more interesting the rule is. Let b_k be the conjunction of $L-1$ of L conditions excluding $a_{i_k} = v_{i_k}$,

$$b_k = \left(a_{i_1} = v_{i_1} \wedge \cdots \wedge a_{i_{k-1}} = v_{i_{k-1}} \wedge a_{i_{k+1}} = v_{i_{k+1}} \wedge \cdots \wedge a_{i_L} = v_{i_L}\right).$$

Assuming independence of $a_{i_k} = v_{i_k}$ and b_k in a whole instance space and in class c, we can predict accuracy of the rule $R2$ from accuracy of a pair of more general rules, $a_{i_k} = v_{i_k} \rightarrow c$ and $b_k \rightarrow c$.

$$\begin{aligned} P(c|a_{i_k} = v_{i_k} \wedge b_k) &= \frac{P(a_{i_k} = v_{i_k} \wedge b_k|c)P(c)}{P(a_{i_k} = v_{i_k} \wedge b_k)} \\ &= \frac{P(c|a_{i_k} = v_{i_k})P(c|b_k)}{P(c)}. \end{aligned}$$

If the real accuracy of $R2$ is comparable to or lower than this expectation, then $R2$ is a trivial conclusion from the rule pair $a_{i_k} = v_{i_k} \rightarrow c$ and $b_k \rightarrow c$, and $R2$ is not interesting at all. We require an interesting rule is more accurate than expected from more general rules shown above for any k.

$$1 \leq \forall k \leq L,\ P(c|a_{i_1} = v_{i_1} \wedge \cdots \wedge a_{i_L} = v_{i_L}) > \frac{p(c|a_{i_k} = v_{i_k})P(c|b_k)}{P(c)}, \quad (1)$$

We also require that accuracy of $R2$ is higher than that of $b_k \rightarrow c$. All instances covered by $R2$ are also covered by $b_k \rightarrow c$ and if we already know the rule $b_k \rightarrow c$ and its accuracy is higher than $R2$, then $R2$ is useless to classify instances into c. Then, we require

$$1 \leq \forall k \leq L,\ P(c|a_{i_1} = v_{i_1} \wedge \cdots \wedge a_{i_L} = v_{i_L}) > P(c|b_k). \quad (2)$$

We only compare accuracy of the rule with more general rules with $L-1$ conditions but don't compare with other general rules with $L-2$ or smaller number of conditions. This is because the difference of accuracy from neighboring regions violating only one condition, is more important than the difference from far regions violating many conditions.

The constraints (1) and (2) give lower bounds of accuracy and we define interestingness of a primitive rule $R2$, $I_{rule}(R2)$, as a margin of its accuracy to satisfy the constraints (1) and (2) as follows.

$$I_{rule}(R2) = \frac{acc(R2) - \max_{1 \leq k \leq L}\left(\max\left(P(c|b_k), \frac{P(c|a_{i_k}=v_{i_k})P(c|b_k)}{P(c)}\right)\right)}{1 - P(c)},$$

where

$$\begin{aligned} &acc(R2) = p(c|a_{i_1} = v_{i_1} \wedge \cdots \wedge a_{i_L} = v_{i_L}), \\ &b_k = \left(a_{i_1} = v_{i_1} \wedge \cdots \wedge a_{i_{k-1}} = v_{i_{k-1}} \wedge a_{i_{k+1}} = v_{i_{k+1}} \wedge \cdots \wedge a_{i_L} = v_{i_L}\right). \end{aligned}$$

The denominator, $1 - P(c)$, is a normalization factor and $I_{rule}(R2)$ becomes 1 when $acc(R2) = 1$ and both of $a_{i_k} = v_{i_k}$ and b_k are independent of c, i.e. $P(c|a_{i_k} = v_{i_k}) = P(c|b_k) = p(c)$.

2.3 Interestingness with Grouping Attribute Values

We can use I_{rule} defined in the previous subsection to evaluate classification rules with grouping attribute values by replacing $a_{i_k} = v_{i_k}$ with $a_{i_k} \in D_{i_k}$, but this sometimes gives large scores to inappropriate rules. Let us show an example in mushroom database why I_{rule} can not be directly applied to evaluate rules with grouping. The next rule is an example with high estimation with respect to I_{rule}.

$$R3\!:\!cap\_color \in \{brown, red\} \wedge stalk\_root \in \{bulbous, rooted\} \rightarrow edible\!:\!100\%.$$

The probabilites required to calculate $I_{rule}(R3)$ are

$$P\,(edible|cap\_color \in \{brown, red\}) = 0.50,$$
$$P\,(edible|stalk\_root \in \{bulbous, rooted\}) = 0.53,$$
$$P\,(edible) = 0.52,$$

and $I_{rule}(R3)$ becomes 0.98. It looks like an exceptional rule because each condition in its body has no correlation with *edible* but only edible mushrooms satisfy the conjunction of them. However, by exploring relationship between *edible* and each attribute value, we can find the following probability.

$$P(edible|stalk\_root = rooted) = 1.00$$

This means that the condition on *cap_color* is redundant for mushrooms with $stalk\_root = rooted$ and $R3$ is not interesting at all for classifying them. Instead, we prefer the following rule that prohibits $stalk\_root = rooted$ and increases the probability of *edible* for all of covered instances compared with the rules whose bodies are one of two conditions.

$$R4 : cap\_color \in \{brown, red\} \wedge stalk\_root \in \{bulbous\} \rightarrow edible : 100\%$$

In this rule, each attribute value correlates with edible as follows and none of them has strong relationship between edible.

$$P\,(edible|cap\_color = brown) = 0.55,$$
$$P\,(edible|cap\_color = red) = 0.42,$$
$$P\,(edible|stalk\_root = bulbous) = 0.51.$$

The problem of I_{rule} is that it evaluates a rule based on only each condition in a body but doesn't concern each attribute value allowed in each condition. To resolve this problem, we introduce interestingness of an attribute value and evaluate how allowing the value contributes to the interestingness of the rule.

Formally, we define the interestingness of an attribute value $a_{i_k} = v$ in a rule $a_{i_1} \in D_{i_1} \wedge \cdots \wedge a_{i_L} \in D_{i_L} \rightarrow c$ as follows.

$$I_{value}(a_{i_k} = v,\ a_{i_1} \in D_{i_1} \wedge \cdots \wedge a_{i_L} \in D_{i_L} \rightarrow c)$$
$$= \max_{v_{i_h} \in D_{i_h}, (h \neq k)} I_{rule}(a_{i_1} = v_{i_1} \wedge \cdots \wedge a_{i_k} = v \wedge \cdots \wedge a_{i_L} = v_{i_L} \rightarrow c).$$

For example, we use the following four primitive rules to evaluate attribute values in $R3$.

$$cap\_color = brown \wedge stalk\_root = bulbous \rightarrow edible\ (I_{rule} = 0.93),$$
$$cap\_color = brown \wedge stalk\_root = rooted \rightarrow edible\ (I_{rule} = 0.00),$$
$$cap\_color = red \wedge stalk\_root = bulbous \rightarrow edible\ (I_{rule} = 1.02),$$
$$cap\_color = red \wedge stalk\_root = rooted \rightarrow edible\ (I_{rule} = 0.00).$$

$Stalk\_root = bulbous$ appears in the first and the third rule and its interestingness is 1.02. Instead, $stalk\_root = rooted$ appears in the second and the last rule and $I_{value}(stalk\_root = rooted, R3) = 0$.

Each instance covered by the original rule is covered by exactly one primitive rule and high score of $I_{value}(a = v, R)$ means there is at least one instance with $a = v$ for which R works better than expected from more general rules. In opposite, $I_{value}(a = v, R)$ becomes low when more general rule can classify the instances with $a = v$ with comparable or higher accuracy.

When extracting a rule whose interestingness is greater than or equals to a certain lower bound, LB, we require not only $I_{rule}(R) \geq LB$ but also $I_{value}(a = v, R) \geq LB$ for each pair of an attribute and its value allowed in the body of the rule. To satisfy the above requirement, we modify the interestingness of rules as follows.

$$IG_{rule}(R1) = \min\left(I_{rule}(R1), \min_{1 \leq k \leq L, v \in D_{i_k}} I_{value}(a_{i_k} = v, R1)\right). \quad (3)$$

Clearly, $IG_{rule}(R)$ coincides with $I_{rule}(R)$ for a primitive rule R that allows exactly one value for each attribute appeared in its body.

In the example of mushroom, $P(edible|stalk\_root = rooted) = 1$ leads I_{value} $(stalk\_root = rooted, R3) = 0$. Then $IG_{rule}(R3)$ becomes 0 and our new criterion of interestingness, IG_{rule}, judges $R3$ is not interesting at all. In contrast, all attribute values in $R4$ get large scores of I_{value} and IG_{rule} judges $R4$ is quite interesting.

3 DIG: Efficient Rule Discovery Algorithm

3.1 Discovery Algorithm

This section gives our discovery algorithm, DIG(Discover Interesting rules with Grouping attribute values). We first assume all attributes are nominal and there is no missing attribute value. We discuss how to deal with numeric attributes and missing attribute values at the end of this section. Inputs of the algorithm are the number of attributes in bodies of rules, L, a lower bound of their support, LB_{sup}, a lower bound of accuracy, LB_{acc}, and a lower bound of interestingness, LB_{ig}. To simplify the discussion, we fix the number of attributes appeared in rules' bodies. We iterate to apply the algorithm to extract a set of rules with different number of attributes in their bodies. An output is a set of rules that

include exactly L attributes in their bodies and satisfy the given constraints on their support, accuracy and interestingness. To avoid extracting similar rules many times, we restrict the algorithm to extract at most one rule for each pair of a class label (head of a rule) and a combination of L attributes. If there are plural rules that share a same label and a same attribute set, then the algorithm selects one of them with a user-defined objective function.

One approach to discover interesting rules is generating all rules with sufficiently large support by apriori[1] like algorithms and filtering out rules that violate the constraints on accuracy and interestingness. This approach is practical to extract only primitive rules but may be too unefficient to extract rules with grouping attribute values. This is because grouping attribute values drastically increases the number of possible rules and the first step, enumerating rules with large support, requires huge time even when L is small. For example, when extracting a rule with a given three attributes with ten possible values, the number of primitive rules is only 10^3 but the number of rules with value grouping becomes $(2^{10}-2)^3 \simeq 10^9$ and the constraint of minimum support usually rejects only the rules that accept a few values for each attribute in their bodies. However, we don't need to enumerate all rules with large support to extract rules with large interestingness. Because of the definition of IG_{rule}, the constraint of minimum interestingness

$$IG_{rule}(a_{i_1} \in D_{i_1} \wedge \cdots \wedge a_{i_L} \in D_{i_L} \rightarrow c) \geq LB_{ig}$$

requires

$$1 \leq \forall k \leq L,\ \forall v \in D_{i_k},$$
$$\exists (v_{i_1}, \ldots, v_{i_{k-1}}, v_{i_{k+1}}, \ldots, v_{i_L}) \in D_{i_1} \times \cdots \times D_{i_{k-1}} \times D_{i_{k+1}} \times \cdots \times D_{i_L},\ \text{s.t.}$$
$$I_{rule}\Big(\Big(\bigwedge_{h \neq k} a_{i_h} = v_{i_h}\Big) \wedge a_{i_k} = v_{i_k} \rightarrow c\Big) \geq LB_{ig}.$$

This property leads the following efficient algorithm that first enumerates primitive rules with large interestingness and generates rules with value grouping by combining the interesting primitive rules. This approach can effectively prune rules with grouping that can't satisfy the constraint on interestingness and can reduce search space drastically compared with enumerating rules with large support first.

Figure 1 shows a pseudo-code of DIG. For each combination of attributes, DIG first counts a class distribution of instnaces in each combination of attribute values to evaluate interestingness of all of possible primitive rules. After that, DIG selects a set S of primitive rules whose interestingness is greater than or equals to the lower bound LB_{ig}. For each subset S' of S, DIG generates a rule R whose body permits all attribute values allowed in at least one primitive rule in S'. DIG evaluates its support, accuracy and IG_{rule} of R. If these values are greater than or equal to the corresponding lower bounds and R is better than the current best rule, R_{best}, with respect to the given objective function, DIG updates R_{best}. After checking all subsets of S, DIG adds R_{best} to a set of

Procedure DIG(*Training*, L, LB_{sup}, LB_{acc}, LB_{ig}, f_{obj})
Inputs:
Training set, *Training*, body length, L,
lower bounds of support, accuracy and interestingness, LB_{sup}, LB_{acc}, LB_{ig}
and an objective function, f_{obj}.
Output:
Set of interesting rules, *Rules*.

$Rules := \emptyset$.
Foreach combination of L attributes $(a_{i_1}, \ldots, a_{i_L})$,
Count class distribution for each combination of values $(v_{i_1}, \ldots, v_{i_L})$ of attributes $(a_{i_1}, \ldots, a_{i_L})$.
Foreach class c
R_{best} :=*undefined*.
Foreach combination of values $(v_{i_1}, \ldots, v_{i_L})$,
Evaluate a primitive rule
$a_{i_1} = v_{i_1} \wedge \cdots \wedge a_{i_L} = v_{i_L} \rightarrow c$.
$S := \{r | r$ *is primitive* $\wedge I_{rule}(r) \geq LB_{ig}\}$.
Foreach $S' \subseteq S$
$D_{i_k} := \{v | \exists r \in S', r$ *permits* $a_{i_k} = v\}, k = 1, \ldots, L$.
$R := a_{i_1} \in D_{i_1}, \wedge \cdots \wedge a_{i_L} \in D_{i_L} \rightarrow c$.
If $Support(R) \geq LB_{sup} \wedge Acc(R) \geq LB_{acc} \wedge IG_{rule}(R) \geq LB_{ig}$ **then**
If R_{best} =*undefined* or $f_{obj}(R) > f_{obj}(R_{best})$ **then**
$R_{best} := R$.
If $R_{best} \neq$*undefined* **then**
$Rules := Rules \cup \{R_{best}\}$.
Rank rules in *Rules* with respect to f_{obj}.
Return*Rules*.

Fig. 1. Pseudo-code of DIG

discovered rules. DIG iterates this procedure for all combinations of L attributes. As discussed above, a rule with $IG_{rule} \geq LB_{ig}$ allows only attribute values appeared in at least one primitive rules with $I_{rule} \geq LB_{ig}$ and this algorithm can explore all rules with grouping values whose interestingness is greater than or equals to LB_{ig}.

3.2 Time Complexity

The most time consuming process in DIG is counting a class distribution of instances in each value pattern to evaluate primitive rules. Because DIG has to count distributions for all combinations of L attributes, the time complexity of this process is $O\left(\binom{M}{L} \cdot N \cdot L\right)$, where M is the number of attributes and N is the number of instances. This complexity depends on N linearly and DIG can extract rules from large number of instances. Instead, the complexity rapidly increases with L and DIG can't extract complex rules with many attributes in their bodies. However, simple rules with small number of attributes are some-

Table 1. Summary of the experiments.

	classes	atts	instances	L	LB_{ig}	extracted rules	cpu time(sec)
mushroom	2	22	8,124	2	0.8	23	5
				3	0.8	10	53
				4	0.8	0	354
satimage	6	36	6,435	2	0.8	12	12
				3	0.8	0	243
				4	0.8	0	4,189
letter	26	16	20,000	2	0.8	16	10
				3	0.8	2	158
				4	0.8	0	2,155

times more important for knowledge discovery than complex rules because we can easily understand the meaning of them.

3.3 Numeric Attributes and Missing Attribute Values

The current version of DIG assumes all attributes are nominal. It can't deal with numeric attributes directly and requires discretization [3] of numeric attributes before discovery process. The only difference between symbolic attributes and discretized attributes is how to generate groups of values as conditions in rules. For a discretized attribute, DIG only permits a value group that represents one interval, i.e. a set of neighboring values. For example, if values of a numeric attribute a_i are discretized into five values, 1, 2, 3, 4 and 5, then DIG permits conditions such as $a_i \in \{1,2\}$ and $a_i \in \{2,3,4\}$, but rejects $a_i \in \{1,4\}$.

Databases from practical domain sometimes involve instances in which values of some attributes are unknown. For each combination of attributes, DIG works with the instances in which values of all of the selected attributes are known. For example, DIG ignores an instance whose value of a_1 is unknown when extracting a rule with an attribute set $\{a_1, a_2\}$ but uses the instance when extracting a rule with $\{a_2, a_3\}$ if the values for a_2 and a_3 are known.

4 Experiments

This section reports the experimental results of DIG on three databases from UCI repository[2], mushroom, satimage and letter recognition. We selected these databases because they involve relatively many instances. In satimage, all attributes are numeric and we discretized each of them into five intervals with same population. Letter database is also represented by numeric attributes but the attributes are already discretized into integers from 0 to 15 and we used them with no change.

For all databases, we set $LB_{acc} = 0.9$ and $L = 2, 3$ and 4, i.e. DIG extracted rules with 90% or higher accuracy that involve 2 to 4 attributes in the bodies. We set minimum support, LB_{sup} at 1% for mushroom and satimage, but used

rule 1: $stalk\_shape = tapering \wedge ring\_type = pendant \wedge habitat = grasses \rightarrow poisonous.$
$support = 1.8\%, accuracy = 100\%, IG_{rule} = 1.30$
$P(poisonous) = 0.48$
$P(poisonous|stalk\_shape = tapering) = 0.44$
$P(poisonous|ring\_type = pendant) = 0.21$
$P(poisonous|habitat = grasses) = 0.34$
$P(poisonous|stalk\_shape = tapering \wedge ring\_type = pendant) = 0.13$
$P(poisonous|stalk\_shape = tapering \wedge habitat = grasses) = 0.16$
$P(poisonous|ring\_type = pendant \wedge habitat = grasses) = 0.33$

(a)Rules extracted from mushroom database.

rule 2: $a_{32} \leq 17 \wedge 34 \leq a_{33} \leq 61 \rightarrow class = 0.$
$support = 6.2\%, accuracy = 98\%, IG_{rule} = 0.80.$
$P(class = 0) = 0.24$
$P(class = 0|a_{32} \leq 17) = 0.31$
$P(class = 0|34 \leq a_{33} \leq 61) = 0.26$

(b)Rules extracted from satimage database.

rule 3: $width \leq 3 \wedge 8 \leq height \leq 9 \rightarrow I.$
$support = 0.4\%, accuracy = 100\%, IG_{rule} = 0.80.$
$P(I) = 0.038$
$P(I|width \leq 3) = 0.213$
$P(I|8 \leq height \leq 9) = 0.037$

(c)Rules extracted from letter database.

Fig. 2. Rules extracted by DIG

0.1% for letter database because this database involves 26 classes and 1% of all instances corresponds to about a quarter of instances in each class. This ratio is too large to discover exceptional patterns. We also fixed the lower bound of interestingness, LB_{ig}, as 0.8. As an objective function, f_{obj}, we applied support of rules and extracted interesting rules covering as many instances as possible. We executed all experiments on sun workstation with 300MHz ultra-sparcII.

Table 1 shows the domain characteristics of the databases and a summery of experimental results, the number of extracted rules and cpu time for extraction. In this table, we can observe that the number of rules with large score of IG_{rule} decreases with L and no rule with $L = 4$ satisfies $IG_{rule} \geq 0.8$ in all databases. A rule with L conditions takes high score with respect to IG_{rule} only when any conjunction of $L-1$ conditions cover many negative instances but the remaining condition can reject most of the negative instances. In natural domain, such a hidden relationship between attributes and a class is very rare when L is large.

Figure 2 shows examples of rules extracted by DIG and probabilities related to the rules. The first rule involves three conditions in its body. Each condition and each pair of the conditions negatively relate to the conclusion, *poisonous*, but the rule shows that the conjunction of the three conditions covers poisonous

mushrooms only. It is not clear whether the rules in the figure are really useful and interesting for the users of discovery systems, but they are at least quite interesting from statistical viewpoint.

In all databases, DIG worked sufficiently fast when $L \leq 3$ and required at most four minutes in satimage. DIG also required practical time when $L = 4$, about 40 minutes in letter recognition and 70 minutes in satimage. The time complexity of DIG depends on L exponentially and DIG may require huge time when $L \geq 5$. However, this is not a critical disadvantage because rules with large interestingness are usually extracted with small L as shown in Table 1.

5 Summary

This paper discussed what kind of classification rules should be extracted by knowledge discovery systems and proposed a new criterion of interestingness of rules that evaluates a rule by comparing its accuracy with those of more general rules. In addition, we pointed out the necessity of evaluation of each attribute value allowed in a body of a rule to evaluate the rule correctly. We also proposed a new discovery algorithm, DIG, to extract interesting rules with respect to the criterion. We applied DIG to three databases and showed DIG could discover interesting rules in practical time.

References

1. Agrawal, R., Mannila, H., Srikant, R. Toivonen, H. and Verkamo, A. I.: Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*, ed. Fayyad, U. M. et al., AAAI Press(1996) 307–328.
2. Blake, C., Keogh, E. and Merz, C. J.: UCI Repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html(1999).
3. Fayyad, U. M. and Irani, K. B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence(IJCAI-93)*, AAAI Press(1993) 1022–1027.
4. Guyon, I., Matic, N. and Vapnik, V.: Discovering Informative Patterns and Data Cleaning. In *Advances in Knowledge Discovery and Data Mining*, ed. Fayyad, U. M. et al., AAAI Press(1996) 181–203.
5. Kamber, M. and Shinghal, R. Evaluating the Interestingness of Characteristic Rules. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining(KDD-96)*. AAAI Press(1996) 263-266.
6. Piatetsky-Shapiro, G.: Discovery, Analysis and Presentation of Strong Rules. In *Knowledge Discovery in Databases*, ed. Piatetsky-Shapiro, G. and Frawley, W. J., AAAI Press(1991) 229–248.
7. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers(1993).
8. Silverschatz, A. and Tuzhilin, A.: What Makes Patterns Interesting in Knowledge Dsicovery Systems. *IEEE Transactions on Knowledge and Data Engineering* Vol.8, No.6(1996), 970–974.

9. Smyth, P. and Goodman, R. M.: Rule Induction Using Information Theory. In *Knowledge Discovery in Databases*, ed. Piatetsky-Shapiro, G. and Frawley, W. J., AAAI Press(1991) 160–176.
10. Suzuki, E.: Autonomous Discovery of Reliable Exception Rules. In *Proceedings: The third International Conference on Knowledge Discovery and Data Mining(KDD-97)*, AAAI Press(1997) 259–262.
11. Srikant, R., Vu, Q. and Agrawal, R.: Mining Association Rules with Item Constraints. In *Proceedings: The third International Conference on Knowledge Discovery and Data Mining(KDD-97)*, AAAI Press(1997) 67–73.

Performance Controlled Data Reduction for Knowledge Discovery in Distributed Databases

Slobodan Vucetic and Zoran Obradovic

School of Electrical Engineering and Computer Science
Washington State University, Pullman, WA 99164-2752, USA
{svucetic, zoran}@eecs.wsu.edu

Abstract. The objective of data reduction is to obtain a compact representation of a large data set to facilitate repeated use of non-redundant information with complex and slow learning algorithms and to allow efficient data transfer and storage. For a user-controllable allowed accuracy loss we propose an effective data reduction procedure based on guided sampling for identifying a minimal size representative subset, followed by a model-sensitivity analysis for determining an appropriate compression level for each attribute. Experiments were performed on 3 large data sets and, depending on an allowed accuracy loss margin ranging from 1% to 5% of the ideal generalization, the achieved compression rates ranged between 95 and 12,500 times. These results indicate that transferring reduced data sets from multiple locations to a centralized site for an efficient and accurate knowledge discovery might often be possible in practice.

Keywords: data reduction, data compression, sensitivity analysis, distributed databases, neural networks, learning curve

1 Introduction

An important knowledge discovery problem is to establish a reasonable upper bound on the size of a data set needed for an accurate and efficient analysis. For example, for many applications increasing the data set size 10 times for a possible accuracy gain of 1% can not justify huge additional computational costs. Also, overly large training data sets can result in increasingly complex models that do not generalize well [8].

Reducing large data sets into more compact representative subsets while retaining essentially the same extractable knowledge could speed up learning and reduce storage requirements. In addition, it could allow application of more powerful but slower modeling algorithms (e.g. neural networks) as attractive alternatives for discovering more interesting knowledge from data.

Data reduction can be extremely helpful for data mining on large distributed data sets where one of the more successful current approaches is learning local models at each data site, and combining them in a meta-model [11]. The advantage of meta-modeling is that learning local models and integrating them is computationally much more efficient than moving large amounts of data into a centralized memory for learning global models. However, this sub-optimal heuristic assumes similarity

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 29-39, 2000.
© Springer-Verlag Berlin Heidelberg 2000

between local data sets and it is not clear how to successfully combine local models learned on data with different distributions and not identical sets of attributes.

A centralized approach of transferring all data to a common location escapes the sub-optimality problems of local models combination, but is often infeasible in practice due to a limited communication bandwidth among sites. Reducing data sets by several orders of magnitude and without much loss of extractable information could speed up the data transfer for a more efficient and a more accurate centralized learning and knowledge extraction. Here, for user-specified allowed accuracy loss we propose an effective data reduction procedure based on applying a guided sampling to identify a minimal size representative sample followed by a model-sensitivity analysis to determine an appropriate compression level for each attribute.

The problem addressed in this study is more formally defined in Section 2, the proposed data reduction procedure is described in Sections 3-4 and an application to 3 large data sets is reported in Section 5.

2 Definitions and Problem Description

To properly describe the goal of data reduction and to explain the proposed approach, some definitions will be given first. The definitions apply to regression and classification problems solved by learning algorithms minimizing least square error, including linear models and feedforward neural networks.

Definitions

Given a data set with N examples, each represented by a set of K attributes, $\mathbf{x}=\{x_1,\ldots,x_K\}$, and the corresponding target y, we denote the underlying relationship as $y = E[y|\mathbf{x}]+\varepsilon$, where ε is an additive error term. For regression problems the target is usually a single number, while for L-class classification problems it is usually an L-dimensional vector. We define the *reduced data set* as any data set obtained from the given one by (1) reduction of the number of examples called *down-sampling*, or/and (2) quantization of its attributes and targets. The *length* of a data set is defined as the number of bits needed for its representation. *Compression* rate *C* equals the ratio between the bitwise length of the original data set and the bitwise length of the reduced data set.

Assuming a parametric learning algorithm, by $f(\mathbf{x}; \beta(n))$ we denote a predictor learned on n examples and we measure its performance by the mean squared error (MSE) defined as $MSE(\beta)=E\{[y- f(\mathbf{x}; \beta)]^2\}$, where β is the set of the model parameters. If a predictor is learned on a reduced, instead of the original, data set some increase in the MSE is to be expected. The total *relative MSE increase*, $MSE(\beta_q(n))/MSE(\beta(N))$, where $\beta_q(n)$ are estimators of parameters from a model learned on a down-sampled data set with quantized attributes, is the product of relative MSE increases due to down-sampling $MSE(\beta(n))/MSE(\beta(N))$ and quantization $MSE(\beta_q(n))/MSE(\beta(n))$. Throughout the text we denote the relative MSE increase as $(1+\alpha)$, and call α the *loss margin*.

Problem Description

Our goal is to obtain a minimal length reduced data set that, using the same learning algorithm, allows extraction of the same knowledge reduced for at most a loss margin α. To achieve this goal we propose two successive phases: (1) reduce the sample size from N to n_{min} allowing loss margin α_D and achieving compression $C_D=N/n_{min}$, and (2) perform proper quantization of attributes of a down-sampled data set allowing loss margin α_Q, followed by Huffman coding [5] of discretized attributes and achieving compression C_Q. Assuming that total loss margin α is close to zero, it follows that $\alpha \approx \alpha_D+\alpha_C$ with an achieved total compression $C = C_D C_Q$. To keep the presentation simple, we will assume that $\alpha_D = \alpha_C = \alpha/2$, and will skip the optimization of α_D and α_C for the maximum achievable total compression.

The motivation for data reduction is obtaining the compact representation of a data set that would facilitate its efficient repeated use with complex learning algorithms and allow its efficient transfer and storage. All these features are highly desirable for data mining on distributed databases. In this framework, local computing time needed for data reduction would not be a critical requirement, since this effort would be rewarded multifold. Nevertheless, for large data sets, the whole data reduction effort has comparable or even lower computational time as compared to building a single model on a whole data set. In the following two sections we separately describe procedures for down-sampling and quantization and compression.

3 Identifying a Minimum Representative Sample

3.1 Down-Sampling for Fast and Stable Algorithms

The *learning curve* for least squares estimators shows the average MSE dependence on the size n of a sample used for designing estimators. This curve can be divided into an *initial region* characterized by the fast drop of the MSE with increasing sample size and a *convergence region* where addition of new samples is not likely to significantly improve prediction (see Fig. 1). The learning curve is the result of complex relationships between data characteristics and a learning algorithm. Therefore its shape needs to be determined by experimentation with an objective of identifying size n_{min} of a minimum representative sample needed to achieve an approximation of the optimal average MSE within a specific *loss margin* α_D.

An asymptotic analysis based on the law of large numbers and the central limit theorem [4] can help in successful modeling of a learning curve. According to the asymptotic normality theorem for nonlinear least squares, estimation error is asymptotically normal under certain fairly general conditions. Asymptotic normality means that $n^{1/2}(\beta(n)-\beta^*)$ tends in distribution to a normal distribution with mean zero and finite covariance matrix, where $\beta(n)$ is an n-sample based estimate of the true parameter vector β^*. The consequence is that for large n, residuals ε' of the nonlinear estimator $f(x;\beta(n))$ consistently estimate the actual disturbances as $\varepsilon' = \varepsilon + O(n^{-1/2})$. Therefore,

$$\mathrm{MSE}(\beta(n)) = \mathrm{MSE}(\beta^*) + u, \quad u \sim N(O(1/n), O(1/n)), \tag{1}$$

meaning that $\mathrm{MSE}(\beta(n))$ asymptotically tends to the optimum $\mathrm{MSE}(\beta^*)$ as $O(1/n)$, with variance decreasing as $O(1/n)$. Assuming that N corresponds to the convergence region and using (1), modeling of a learning curve to estimate a minimum representative sample size n_{min} can be fairly straightforward.

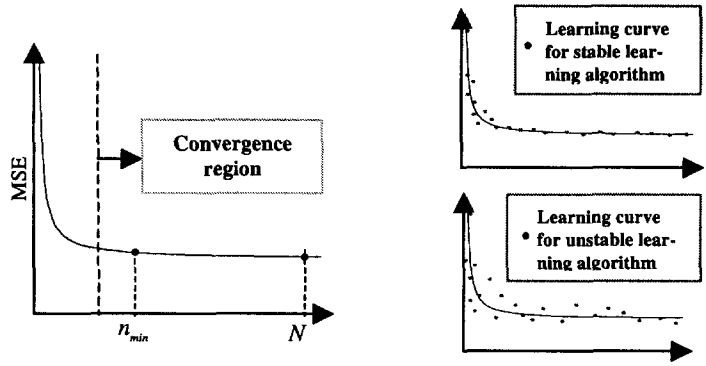


Fig. 1. Learning Curve

A recently proposed progressive sampling procedure [9] can efficiently span the available range of sampling sizes in search for the n_{min}. The technique was developed with an objective of increasing the speed of inductive learning by providing approximately the same accuracy and using significantly smaller sample sizes than available. It was shown that geometrical progressive sampling that starts with a small sample and uses geometrically larger sample sizes until exceeding n_{min} (model accuracy no longer improves) is an asymptotically optimal sampling schedule. We use the idea of progressive sampling with a somewhat different motivation of guiding an efficient search for a minimal sample size needed for achieving an approximation of the optimal average MSE within a specific *loss margin* α_D.

In regression statistical theory it is well known that linear least squares algorithms are the optimal estimators for linear problem solving. They are characterized by fast learning with time complexity $O(n)$ and well-known statistical properties including small variance. The following *DS1* procedure is proposed for identifying n_{min} value for fast and stable models:

- Estimate model on the whole available data set of size N and calculate MSE(N);
- Estimate model on a small sample of size n_0 and calculate MSE(n_0);
- Increase the sampling size from n_i to $n_i \cdot a^i$ until a sample size n_{min} is reached satisfying $\mathrm{MSE}(n_{min}) < (1+\alpha_D)\mathrm{MSE}(N)$.

A direct consequence of the progressive sampling results [9] for models with time complexity $O(n)$ is that the time complexity of this procedure for a=2 is at most twice the time of learning on the whole data set. This procedure might also be used for simple nonlinear algorithms with small variance (e.g. the feedforward neural networks without hidden nodes).

3.2 Down-Sampling Extension for Slower and Unstable Algorithms

Complex nonlinear learning algorithms such as feedforward neural networks with a number of hidden nodes typically have a large variance meaning that their MSE($\beta(n)$) can largely differ over different weight's initial conditions and choice of training data. Using explained *DS1* down-sampling procedure for such learning algorithms could cause significant errors in the estimation of n_{min}. Also, with these algorithms learning time for large N can be so long that the cost of obtaining a benchmark value of MSE(N) is unacceptable.

Using (1) and assuming that N is within a learning curve convergence region down-sampling can be performed by fitting learning curve samples obtained through guided sampling as

$$MSE(n) = \gamma_0 + \gamma_1 / n + \gamma_2 / n^2 + e, \quad e \sim \mathrm{N}(0, O(1/n)), \tag{2}$$

where γ_0 corresponds to an estimate of MSE for an infinitely large dataset, γ_1 to O(1/n) part of (1), and γ_2 to the initial region of a learning curve.

The error variance of the learning curve samples decreases as O(1/n), and so larger confidence should be given to MSE's of estimators learned on larger data samples. Therefore, we apply a weighted least squares algorithm [6] to fit the learning curve by multiplying (2) by $n^{1/2}$ and learning γ's on transformed learning curve samples.

For slower and unstable algorithms we propose the following down-sampling procedure that we will call *DS2*:

- Starting from a sample of size n_0, at iteration i increase sample size to $n_i = n_0 \cdot a^i$ until t-statistics for γ_0 and γ_1 do not exceed $t_{\theta,n-3}$ for some tolerant confidence level θ;
- Repeat until the difference in estimating n_{min} over several successive iterations is sufficiently small:
  - According to estimated γ's and predetermined loss margin, α_0, estimate n_{min} using (2);
  - Select the next sample size n_{i+1} larger then n_{min}. Larger n_{i+1} results in a larger improvement in the estimation of n_{min}, but at increased computational cost. Our heuristic of randomly selecting n_{i+1} within an interval $[n_{min}, 2n_{min}]$ has proven to be a good compromise;
  - Learn a new model on n_{i+1} samples and calculate its MSE($\beta(n_{i+1})$);
- Output the last estimated n_{min} as the minimum representative sample size.

If neural networks are used in the down-sampling procedure, the minimum sample size is selected larger than the estimated value since part of the data should be reserved to validation subset. Our heuristic determines the total representative size as $1.5n_{min}$ such that in all iterations of down-sampling algorithm, $0.5n_i$ samples are being used as a validation set for an early stopping of neural network training.

4 Compression of a Minimum Representative Sample

Storing continuous attributes usually assumes double precision (8 bytes) for an accurate representation. Performing quantization of a continuous variable into a number of bins allows its representation with a finite alphabet of symbols. This allows the use of known compression algorithms [10] to decrease the number of bits needed to represent a given attribute at the price of introducing a certain level of distortion. We employ uniform quantization where the range of a continuous attribute x is partitioned into Q equal subintervals, and all numbers falling in a given interval are represented by its midpoint. Denoting quantizer operation as $x_q=Q(x)$, the *quantization error*, $\varepsilon_q=x_q-x$, can be considered as uniformly distributed in a range $[-q/2, q/2]$, where $q = (x_{max} - x_{min})/Q$ is the *quantization interval*. Therefore, quantization error variance equals $q^2/12$.

Given a data vector $\{x^1, \ldots, x^n\}$ over a finite Q-ary alphabet $=\{a_1, \ldots, a_Q\}$ (obtained by quantization), we apply Huffman coding where more frequent symbols are assigned shorter encoding lengths [5]. This provides the optimal binary representation of each symbol of without any loss of information, such that the output bitstring length $\sum f_i l_i$ is minimized, where f_i is frequency of a_i and l_i is length of its binary representation.

4.1 Model Sensitivity Analysis for Attributes Quantization

In data reduction for knowledge discovery, preserving fidelity of all the attributes is not important by itself. A better goal is preserving the fidelity of the prediction model learned using these attributes as measured by a loss margin α_Q. With this goal, less sensitive attributes can be allowed higher distortion and, therefore, be quantized to lower resolution by using larger quantization intervals. To estimate the influence of attribute's quantization on model predictions we propose the following sensitivity analysis of a model obtained on the down-sampled data set. The outcome of this analysis allows deducing proper relative quantization levels for all attributes resulting in an efficient quantization procedure.

For a small quantization interval q_i the function $f(\mathbf{x}_{qi}, \beta(n_{min}))$ can be approximated as

$$f\left(\mathbf{x}_{qi}, \beta(n_{\min})\right) \approx f\left(\mathbf{x}, \beta(n_{\min})\right) + \frac{\partial f\left(\mathbf{x}, \beta(n_{\min})\right)}{\partial x_i} \varepsilon_{qi}, \tag{3}$$

where ε_{qi} is quantization error with a uniform distribution over $[-q/2, q/2]$ and $\mathbf{x}_{qi}$ denotes an input vector with quantized attribute X_i,.

From (3) the relative MSE increase due to quantization of attribute X_i is

$$RMSE_Q(q_i) = E\left\{\left(f(x_{qi}, \beta) - f(x, \beta)\right)^2\right\} \approx \frac{q_i^2}{12} \int_{D_{xi}} \left(\frac{\partial f(x, \beta)}{\partial x_i}\right)^2 dp(x_i), \tag{4}$$

where $p(x_i)$ is the distribution of attribute X_i.

The integral in (4) could be approximated as

$$\int_{D_{xi}} \left(\frac{\partial f(x)}{\partial x_i} \right)^2 dp(x_i) \approx \frac{1}{n_{\min}} \sum_{j=1}^{n_{\min}} \left(\frac{f\left(x^j + \delta x_i\right) - f\left(x^j\right)}{\delta x_i} \right)^2, \quad (5)$$

where δx_i is a small number (we used $\delta x_i = \text{std}(x_i)/1000$).

By X_j we denote the most important attribute with the largest $RMSE_Q(\delta x_i)$, $i=1,\ldots,K$. Let us quantize attribute X_j such that the number of quantization intervals inside a standard deviation of X_j is M_j where the constant M_j is called the *quantization density*. Quantization densities of other attributes are selected such that losses of all the attributes are the same. These densities can be computed from (4) as

$$M_i = M_j \sqrt{\frac{RMSE_Q(\delta x_i)}{RMSE_Q(\delta x_j)}} = M_j \xi_i, \quad (6)$$

where $\xi_i \leq 1$ is a *correction factor* that measures the relative importance of the attributes and is the key parameter allowing an efficient quantization.

4.2 Quantization Procedure for Attributes and Target

If an attribute is nominal or already has discrete values it can be directly compressed by Huffman coding. If it is continuous, its quantization can greatly improve compression without loss of relevant knowledge.

Using correction factors ξ_i, a proper M_j needs to be estimated to satisfy a quantization loss margin α_Q. For a given M_j we calculate M_i, $i=1,\ldots,K$, to quantize all K attributes. We denote a quantized version of an example $\mathbf{x}$ as $\mathbf{x}_q$.

Starting from a small M_j we should estimate true loss as $\text{MSE}(\beta_q(n_{min}))/\text{MSE}(\beta(n_{min}))$ and should gradually increase M_j until this ratio drops below α_Q. At each iteration of M_j this requires training a new model $f(\mathbf{x},\beta_q(n_{min}))$ with quantized attributes which could be computationally expensive. However, our experience indicates that estimating $E\{[y-f(\mathbf{x}_q,\beta(n_{min}))]^2\}$ leads to a slightly pessimistic estimation of $\text{MSE}(\beta_q(n_{min}))$ which can be done by using an already existing model $f(\mathbf{x},\beta(n_{min}))$ from a down-sampling phase. Hence, to improve speed without much loss of accuracy we use $E\{[y-f(\mathbf{x}_q,\beta(n_{min}))]^2\}$ in the quantization procedure. When a proper size M_j is found, quantization densities for all continuous attributes M_i are calculated from (6) and quantized accordingly.

For classification, target compression can be very successful. If a target is continuous, we propose a representation with single or double precision, since for knowledge discovery the accuracy of target is usually more important then the accuracy of attributes. Finally, after a proper quantization of continuous attributes Huffman coding is applied to obtain an optimally compressed data set. Along with the compressed data set, a small table containing the key for Huffman decoding is saved.

5 Experimental Results

To illustrate the potential of the proposed data reduction procedure we performed experiments on 3 large data sets. The first data set corresponds to a simple regression problem, while the remaining two are well-known benchmark classification problems for knowledge discovery algorithms [7].

Normal Distribution Set

We generated a data set consisting of N=100,000 examples with 10 normally distributed attributes, x_i, i=1,...,10, and target y generated as a linear function, $y=\Sigma\beta_i x_i+\varepsilon$ for randomly chosen parameters β_i and the error term being normally distributed and containing 50% of the total variance of y. Assuming standard double precision representation, the total size of this data set is 8.8MB. We chose this set to test our down-sampling procedure *DS1*, and we used an optimal linear estimator with n_0=10, a=1.5, and loss margin set to α={0.01, 0.02, 0.05}. An extremely large compression rate of up to 1,100 times to only 8KB, with minimal model accuracy loss, was achieved as reported in **Table 1**. It is interesting to observe that almost 1/3 of the reduced data set length was used for the target representation since we intentionally decided not to compress targets due to their importance.

Table 1. Data reduction results for normal distribution data set. Here α is the prespecified loss margin, M_j is the quantization density for the most relevant attribute, *loss* is an actual accuracy loss when using reduced data for modeling, *RDS* is the reduced dataset size and *C* achieved compression rate. The original double precision representation was 8.8 MB

α	Linear Estimator					Neural Network			
	n_{min}	M_j	*loss*	*RDS*	*C*	$1.5n_{min}$	M_j	*RDS*	*C*
0.01	**1900**	10	0.007	13 KB	**680**	**4220**	8	25 KB	**350**
0.02	**1120**	8	0.009	10 KB	**880**	**2250**	6	19 KB	**460**
0.05	**420**	5	0.033	8 KB	**1100**	**1020**	4	14 KB	**630**

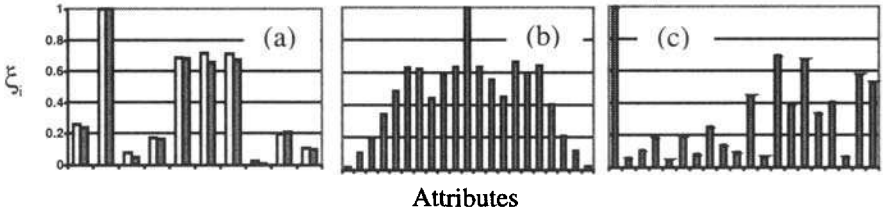


Fig. 2. Correction factors from sensitivity analysis for (a) normal distribution set (left bars are correct and right estimated correction factors), (b) WAVEFORM data set, and (c) COVERTYPE data set

We also used a neural network with 3 hidden nodes in a down-sampling procedure *DS2* to estimate the consequences of a non-optimal choice of the learning algorithm.

For small sample sizes, neural networks tend to overfit the data, and hence, computed n_{min} is significantly larger than for linear estimators. Therefore, the compression rate was slightly smaller than for a linear estimator, but still very large. It could also be noted that in both cases the quantization interval is fairly large, as could be concluded from small value of relative quantization density M_j. The experimentally estimated correction factors for 10 attributes obtained through a sensitivity analysis (right bars at Fig. 2a) were compared to the true values (left bars at Fig. 2a) and it was evident that the sensitivity analysis was extremely successful in proper estimation of attributes importance.

WAVEFORM Data Set

As originally proposed by Breiman [2, 7], we generated 100,000 examples of a data set with 21 continuous attributes and with 3 equally represented classes generated from a combination of 2 of 3 "base" waves. The total size of this data set with double precision was 17.8 MB. In a down-sampling procedure with n_0=100, a=1.5, and loss margin set to α={0.01, 0.02, 0.05} we used neural networks with 5 hidden nodes, 21 inputs and 3 outputs. Observe that the number of examples needed for successful knowledge extraction was significantly higher than in the normal distribution problem as expected for a higher complexity concept. However, for all loss margins the obtained data reduction was still very high (see **Table 2**) while estimated attributes correction factors ξ_i recovered the structure of 3 waveforms hidden in the data (see **Figure 2b**). Our neural network models trained on a reduced data set of length 186KB achieved an average generalization accuracy of 86%, which is identical to the previously reported accuracy using all 17.6 MB of training data.

Table 2. Data reduction results for WAVEFORM data set (notation is same as in Table 1)

α	$1.5n_{min}$	M	*RDS*	C
0.01	**19670**	8	186 KB	**95**
0.02	**10640**	6	89 KB	**200**
0.05	**4580**	4	33 KB	**530**

COVERTYPE Data Set

This is currently one of the largest databases in the UCI Database Repository [7] containing 581,012 examples with 54 attributes and 7 target classes and representing the forest cover type for 30 x 30 meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System [1]. In its raw form it has 75.2 MB, and in the compressed 11.2 MB. Out of 54 attributes, 40 are binary columns representing soil type, 4 are binary columns representing wilderness area, and the remaining 10 are continuous topographical attributes. Seven classes represent forest cover type. Since 40 attributes for just one variable seemed too much for neural network training, we transformed them into 7 new ordered attributes by the following simple „trick". For each of 40 soil types we calculated the relative frequency of each of 7 classes from the available examples. In that way each soil type value was represented as a 7-dimensional vector with values that could be considered continuous and were fit for use with neural networks. The transformed data set had 21 attributes and in the down-

sampling procedure *DS2* with n_0=100, a=1.5, for loss margin set to α={0.01, 0.02, 0.05} we used neural networks with 5 hidden nodes, 21 inputs and 7 outputs. In the quantization procedure we quantized only 10 continuous attributes, while nominal soil type and wilderness area attributes were, together with the target variable, compressed by Huffman coding directly. Data reduction results presented in **Table 3** show that surprisingly large data reduction of several thousands times can be performed without significant knowledge loss and achieving about 70% accuracy as consistent with previous reported results [1].

Table 3. Data reduction results for COVERTYPE data set (notation is same as in Table 1)

α	$1.5n_{min}$	M	*RDS*	C
0.01	**6860**	8	26 KB	**2890**
0.02	**3690**	6	14 KB	**5370**
0.05	**1680**	4	6 KB	**12500**

It should be noted that approximately 1 KB of reduced data set size is used to represent a very informative 40×7 table of relative frequencies for 7 classes on 40 soil types. The estimated attribute correction factors are shown in Fig. 2c. One of the by-products of this sensitivity analysis indicates that the most important attributes for this problem are elevation and soil type, followed by wilderness area attribute.

One of the reasons for such successful reduction of this data set is possibly in its spatial component, and a relatively dense spatial grid (30×30 meters). To better exploit the spatial component of the COVERTYPE data set it would be desirable if positions of examples were also included in the form of x and y coordinates. This would allow the use of the elements of spatial statistics [3] and adjusted learning algorithms [12] for better knowledge extraction.

6 Conclusions

In this paper we proposed a set of procedures aimed at performance-controlled reduction of a given data set by: (1) elimination of redundant training examples, and (2) attributes quantization and compression. The data reduction goal was to obtain a minimal length reduced data set that, using the same learning algorithm, allows extraction of the same knowledge reduced for at most a predetermined loss margin.

Experiments were performed on a large regression and two large classification data sets. An ordinary least squares algorithm and neural networks were used to guide data reduction. Depending on prespecified loss margins of 1% to 5% of full accuracy, the achieved compression rates ranged between 95 and 12,500 times, indicating possible huge benefits for centralized knowledge discovery in distributed databases.

We obtained few other results worth mentioning. The proposed sensitivity analysis proved very successful in ranking the attributes and allowed an efficient compression of continuous attributes. This analysis can be considered separately as a method for soft feature reduction and feature selection that is based directly on their importance for a given learning model. Our results also show that a proper choice of learning

model is important for data reduction and that a reduced data set can be used as a good indicator of the complexity of a learning problem.

The proposed procedure is suited for learning algorithms based on least squares minimization, and could be applied to a range of classification and regression problems. Further work is needed to extend the technique to other learning algorithms.

References

[1] Blackard, J., *Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types*, Ph.D. dissertation, Colorado State University, Fort Collins, 1998.

[2] Breiman, L., Friedman, J., Olshen, R., Stone, C., *Classification and Regression Trees*, The Wadsworth International Group, 1984.

[3] Cressie, N.A.C., *Statistics for Spatial Data,* John Wiley & Sons, Inc., New York, 1993.

[4] Davidson, R., MacKinnon, J., *Estimation and Inference in Econometrics,* Oxford University Press, New York, 1993.

[5] Huffman, D, A Method for the Construction of Minimum Redundancy Codes, *Proc. IRE*, vol. 40, pp 1098-1101, 1952.

[6] Judge, G., Lee, T.C., Hill, C., *Introduction to the Theory and Practice of Econometrics*, John Wiley & Sons, 1988.

[7] Murphy, P.M., Aha, D.W., *UCI Repository of Machine Learning Databases,* Department of Information and Computer Science, University of California, Irvine, CA, 1999.

[8] Oates, T., Jensen, D., Large Datasets Lead to Overly Complex Models: An Explanation and a Solution, *Proc. Fourth Int'l Conf. on Knowledge Discovery and Data Mining*, 1998.

[9] Provost, F., Jensen, D., Oates, T., Efficient Progressive Sampling, *Proc. Fifth Int'l Conf. on Knowledge Discovery and Data Mining*, 1999.

[10] Sayood, K., *Introduction to Data Compression*, Academic Press/Morgan Kaufmann, 1996.

[11] Stolfo, S., Prodromidis, A., Tselepis, S., Lee, W., Fan, D., Chan, P., JAM: Java Agents for Meta-learning over Distributed Databases, *Proc. Third Int'l Conf. on Knowledge Discovery and Data Mining*, 1997.

[12] Vucetic, S., Fiez, T., and Obradovic, Z., A Data Partitioning Scheme for Spatial Regression, *Proc. IEEE/INNS Int'l Conf. on Neural Networks*, No. 348, session 8.1A, 1999.

Minimum Message Length Criterion for Second-Order Polynomial Model Discovery

Grace W Rumantir

School of Computer Science and Software Engineering
Monash University – Clayton Vic 3168 Australia
gwr@csse.monash.edu.au

Abstract. This paper proposes a method based on the Minimum Message Length (MML) Principle for the task of discovering polynomial models up to the second order. The method is compared with a number of other selection criteria in the ability to, in an automated manner, discover a model given the generated data. Of particular interest is the ability of the methods to discover (1) second-order independent variables, (2) independent variables with weak causal relationships with the target variable given a small sample size, and (3) independent variables with weak links to the target variable but strong links from other variables which are not directly linked with the target variable. A common non-backtracking search strategy has been developed and is used with all of the model selection criteria.

Keywords: scientific discovery, automated modelling, second-order polynomial regression

1 Polynomial Model Selection Criteria

Polynomial regression concerns with the task of estimating the value of a target variable from a number of regressors/independent variables. The standardized second-order polynomial regression models considered in this paper typically take the form

$$y_n = \sum_{p=1}^{P} \gamma_p u_{np} + \sum_{p=1}^{P} \sum_{q \geq p}^{P} \gamma_{pq} u_{np} u_{nq} + \epsilon_n \Leftrightarrow y_n = \sum_{k=1}^{K} \beta_k x_{nk} + \epsilon_n \qquad (1)$$

where for each data item n:

y_n	:	target variable	x_{nk}	:	regressor k; $x_{nk} = u_{np}$ or $x_{nk} = u_{np}u_{nq}$; $q \geq p$
u_{np}	:	regressor p	β_k	:	coeff. for regressor k
γ_p	:	coeff. for single regressor	K		$=2P + P!/2!(P-2)!$
γ_{pq}	:	coeff. for compound regressor	ϵ_n	:	noise/residual/error term

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 40–48, 2000.
© Springer-Verlag Berlin Heidelberg 2000

The values of the error term ϵ is assumed to be uncorrelated, normally and independently distributed $\epsilon_n \sim NID(0, \sigma^2)$. For a given set of x_k, this assumption causes y to have the same normal distribution and variance as ϵ, that is, $y_n \sim N(\sum_{k=1}^{K} \beta_k x_{nk}, \sigma^2)$. Given that y is normally distributed, the fact that we can write $\hat{\beta}_k$ as a linear combination of the y values implies that $\hat{\beta}_k$ also has a normal distribution.

Unless the exact relationships between the target variable and the regressors in a problem domain are known, it is necessary to search among all of the potential regressors available for a subset that has the strongest explanatory relationships with the target variable to form a polynomial model to be used for future predictions. The search space can be large especially when the products of variables are also considered (higher-order polynomials).

This work examines some of the most commonly cited penalized-likelihood methods used to compare models with different complexities, along with a Bayesian selection method based on the Minimum Message Length (MML) principle [14] developed for this purpose. The summary of the methods is given in Table 1. The explanation of the methods is given in Section 1.1 and 1.2. The model chosen by any of the methods is claimed to be a parsimonious description of the data at hand, therefore has predictive power for future data. This work tests the robustness of each method in support of this claim.

Previous related studies on the comparison of MML and other methods for the purpose of curve-fitting can be found in [2] and [13]. In this paper, the task is to estimate the order of the regressors of a given polynomial model.

Table 1. Summary of model selection criteria. i is index for sample item ranging from 1 to n and k is the number of variables in a model

Method		Ref.	Objective Function
Minimum Message Length	MML	[14]	$-\log f(x\|\theta) + \frac{1}{2} log\|I(\theta)\| - \log h(\theta) -$ $-\frac{k+1}{2} \log 2\pi + \frac{1}{2} \log(k+1)\pi - 1 - \log h(\nu, \xi, j, t, J, L)$
Minimum Description Length	MDL	[8]	$-\log f(x\|\theta) + \frac{k}{2} \log n + (\frac{k}{2} + 1) + \log k(k+2)$
Corrected AIC	CAICF	[3]	$-\log f(x\|\theta) + \frac{1}{2} log\|I(\theta)\| + k + \frac{1}{k} \log n$
Structured Risk Minimisation	SRM	[12]	$\frac{1}{n} \sum_{i=1}^{n} e_i^2 / (1 - \sqrt{\frac{(k+1)(\log \frac{n}{k+1} + 1) - \log \eta}{n}})$
Stochastic Complexity	SC	[9]	$\frac{n}{2} \log \sum_{i=1}^{n} e_i^2 + \frac{1}{2} \log \|X'X\|$
Akaike's Information Criterion	AIC	[7]	$-2(\log f(x\|\theta) - k)$
Bayesian Information Criterion	BIC	[11]	$-2(\log f(x\|\theta) - \frac{1}{2} k \log n)$
Mallows' C_p	C_p	[6]	$\frac{1}{\hat{\sigma}^2} \frac{1}{n} \sum_{i=1}^{n} e_i^2 - (n - 2k)$
Adjusted Coeff. of Determination	adjR^2	[5]	$1 - (\sum_{i=1}^{n} e_i^2/(n-k))/(\sum_{i=1}^{n} (y_i - \bar{y})^2/(n-1))$
F-to-enter	F-test	[7]	$(\sum_{i=1}^{n} e_i^2)_{old} \geq (\sum_{i=1}^{n} e_i^2)_{new} (1 + F_{enter}/(n - (k+1)))$
F-to-exit			$(\sum_{i=1}^{n} e_i^2)_{new} \geq (\sum_{i=1}^{n} e_i^2)_{old} (1 + F_{exit}/(n - (k+1)))$

1.1 Minimum Message Length Method

From information theory perspective, Minimum Message Length (MML) principle [14] takes the metaphor of sending data over a communication line. Referring to Equation 1, suppose the sender has 2 sets of data, one of the target variable y_n and the other of the regressors x_{nk}. The receiver only has the data set of the regressors x_{nk} and would like to have the data set of the target variable y_n sent. The most expensive way to do this would be to send the encoding of all of the data verbatim. This is equivalent to what is known as a table lookup. The cheapest way would be by first developing a polynomial model in the form of Equation 1 and then sending the optimal encoding of the model and residual data which minimizes the total message length:

$$L = L(\theta) + L(x|\theta) \Leftrightarrow -\log f(\theta|x) = -\log P(\theta) - \log f(x|\theta) \tag{2}$$

The first part of the message length represents the model complexity and the second part represents the goodness of fit of the model into the data. When comparing two models with different complexities, the model with the shorter two-part message length would be chosen. From Bayesian perspective, MML principle states that the best model is the one that yields the highest posterior probability by maximizing the product of the prior probability of the data given the model (Equation 2).

The cost of encoding the model $L(\theta)$ is composed of two parts: that of the model structure (i.e. which combination of variables) L_s and that of the model parameters L_p. Hence the total message length is:

$$L = L_s + L_p + L(x|\theta) \tag{3}$$

In this paper, the cost of encoding the model structure L_s is composed of three parts: that of the set of single regressors, product of regressors and the combination of regressors:

$$L_s = -\log h(\nu, j) - \log h(\xi, l) - \log \frac{1}{\binom{J+L}{j+l}} \tag{4}$$

$h(\nu, j)$ and $h(\xi, l)$ follow geometric series:
$h(\nu, j) = \nu^j(1-\nu)/(1-\nu^{J+1})$ and $h(\xi, l) = \xi^l(1-\xi)/(1-\xi^{L+1})$

where: ν, ξ : probability of choosing single, product of regressors
j, l: single, product of regressors chosen
J, L: single, product of regressors available

It is assumed that the sender and receiver of the message have some prior knowledge/expectation about the possible models $\theta = (\beta_k, \sigma)$, giving a message length $L = -\log h(\theta) = -\log prior(\sigma) prior(\{\beta_k|\sigma\})$. However, the adoption of a discrete message/code string of length L implies that θ itself is regarded as

having a prior probability of e^{-L}, which is discrete. MML principle assigns to θ a prior probability $h(\theta) * v(\theta)$ where $v(\theta)$ is the the volume of a region of the search space which includes θ. As shown in [14] the whole message length is minimized when $v(\theta)$ is chosen to be proportional to $1/\sqrt{|I(\theta)|}$, where $|I(\theta)|$ is the Fisher information matrix.

As in Equation 1, the parameters β_k are assumed to be normal. If uniform density prior (i.e. no preference) is chosen for σ, then following [14], the cost of encoding the model parameters L_p takes the form

$$L_p = -\log prior(\sigma) prior(\{\beta_k|\sigma\}) + \frac{1}{2}\log|I(\theta)| - \frac{1}{2}(k+1)\log 2\pi + \frac{1}{2}log(k+1)\pi - 1$$

$$= \frac{1}{\sigma}\prod_{k=1}^{K}\frac{1}{\alpha\sigma\sqrt{2\pi}}e^{-\beta_k^2/2\alpha^2\sigma^2} + \frac{1}{2}\log|I(\theta)| - \frac{1}{2}(k+1)\log 2\pi + \frac{1}{2}log(k+1)\pi - 1 \quad (5)$$

The term $|I(\theta)| = 2N\sigma^{-2(K+1)}|X'X|$ is the expected Fisher information matrix associated with the real-valued parameters of θ. That is, the determinant of the expected second partial differentials of $L(x|\theta)$ (Equation 6) with respect to the model. The last three terms reflect the effect of quantization of $v(\theta)$ (i.e. the discretization of θ) in forming the optimum code [4, pp.59–61] which results in the increase in the message length.

Finally, the cost of encoding the data given the model $L(x|\theta)$ is simply the likelihood function

$$L(x|\theta) = -\log f(y|\sigma, \{\beta_k\}) = -\log\prod_{n=1}^{N}\frac{1}{\sigma\sqrt{2\pi}}e^{-(y_n-\sum_{k=1}^{K}\beta_k x_{nk})^2/2\sigma^2} \quad (6)$$

1.2 Comparison Between MML and the Other Methods

The methods in Table 1 can broadly be categorized as penalized-likelihood methods. When comparing two models, not only do they consider how a model fits the data, but they also have penalty terms for the more complex model. Hence, the model can be chosen solely from training data. The terms $-logf(x|\theta)$ (Equation 6) or $\sum_{i=1}^{n} e_i^2$ in the equations in Table1 typically measure how a model fits a data set and the rest represents the penalty terms for model complexity. Following [7], F_{enter} and F_{exit} is given as 4.0. Following [12], η is given as 0.125.

Rissanen proposes two methods for implementing minimum encoding, the Minimum Description Length (MDL) principle and the Stochastic Complexity (SC) principle. SC is seen as a refinement of MDL since for large values of n, $X'X$ is approximately proportional to n, hence $|X'X|$ behaves like n^k, and the second term of SC is like the second term of MDL [9]. The term $|X'X|$ is included in $|I(\theta)|$ (Equation 5) for MML which will converge to the same value for large n. It can be concluded that MDL incorporates all the terms of SC. The difference between the model chosen by MDL and MML relies on the values for the third term onwards in each equation.

The penalty term for AIC, k (the number of independent variables) turns out to be the same as Mallows' C_p criterion. BIC is similar to the first two

terms of MDL. [3] sees BIC as AIC altered since $\frac{k}{n} \rightarrow 0$ as $n \rightarrow \infty$. BIC has a bigger penalty term for model complexity than AIC, hence should be less likely to overfit.

As a variation of AIC, [3, p.361] derived CAICF as an estimate of minus the expected entropy, hence does not follow a Bayesian approach. Like MML, CAICF uses the Fisher information matrix as a more precise estimate of the expected log likelikehood.

1.3 Search Algorithm

A non-backtracking search algorithm has been developed to be used as a common search engine for the different stopping criteria. This algorithm starts with an empty model. Variables are either added to or deleted from the model one at a time. A new variable will be added to the model if it results in the best among the other alternative new models based on the selection criterion being used. After every addition to the model, a variable will be searched to be deleted therefrom. A variable will be deleted from the model if it results in the best new model among the initial model and any other models should any other variable has been chosen. Hence in effect, at each stage of the search process, the model chosen would be the best amongst all of the potential models with the same complexity that are possibly chosen by adding or deleting one variable to or from the existing model. The search terminates when there is no more variable to be added which will result in a better model. In case a model selection method overfits the data, a limit in the maximum number of variables that a model can have is imposed to enable the search to terminate in a reasonable amount of time. In this paper, a model can have a maximum of 70 variables.

2 Experimental Design

Three true models as shown in Figure 1, 2 and 3 have been designed for the experiments. Each true model consists of a target variable and a set of single and compound independent variables. Not all of the variables are necessarily directly or at all connected to the target variable. Each value of an independent variable is chosen randomly from a normal distribution $N(0, 1)$. For each model, 6 training and test data sets comprising 500, 1000, 2000, 4000, 6000 and 10000 instances respectively are generated.

The product of two independent variables is calculated from the standardized values of each variable. Each value of the target variable is calculated from the values of all of the independent variables directly linked to it multiplied by the respective link weights plus a noise value which is independently and identically distributed (i.i.d) as Normal $(0, 1)$.

The search engine is presented with the data of the target variable and all of the available independent variables and the possible products of the single variables. The performance criteria for the true model discovery task are whether or not a model selection method manage to select a model with the same set of

variables and corresponding coefficients as those of the true model, as reflected in the following measures:

1. The number of variables selected in the discovered model
2. How close the coefficients of the discovered model are with those of the true model (model error): $1/K * \sum_{k=1}^{K}(\beta_k - \hat{\beta}_k)^2$
3. Model predictive performance (on test data), quantified by
   (a) Root of the mean of the sum of squared deviations: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$
   (b) Coefficient of determination: $R^2 = 1 - (\sum_{i=1}^{n} e_i^2)/\sum_{i=1}^{n}(y_i - \bar{y})^2$

Jacobian Orthogonal Transformation (as opposed to straight Gaussian Elimination) is performed on the covariance matrix of the independent variables for the calculation of model coefficients. This ensures that should multicollinearity among variables exists, it is not reflected in the model coefficients. The normalized variances of the single and product of variables are kept to unity by standardizing the product of the standardized single variables.

3 Results and Discussions

The results of the experiments with artificial data given in Table 2 show that the search engine using model selection methods MML, MDL, CAICF, SRM or SC manages to home into the true models (i.e. all of the variables with direct links to the target variable shown in the number of variables discovered. Due to space constraint, the variables and their coefficients are not shown).

The other methods, namely AIC, BIC, adjR^2 and F-test tend to choose wrong and much more complex models. The fact that the models selected by AIC, BIC, adjR^2 for Model 2 and 3 have 70 variables for all of the sample sizes suggests that the search has been stopped before convergence.

From the performance criteria and the number of variables chosen for Model 1, 2 and 3, it is clear that AIC, BIC, adjR^2 and F-test have *overfitted the training data.* Hence, this implies that in those model selection methods, the penalty for chosing a more complex model is too small compared to the reward of better data fit.

Nonetheless, it has been observed that all of the methods selected some of the significant regressors early on in the search process and assigned relatively large coefficients to them and small coefficients to the variables chosen which do not exist in the true model.

These results suggest that if a model selection procedure is to be fully automated, MML, MDL, CAICF, SRM and SC can reliably converge to the true model (if one exists), or to a reasonably parsimonious model estimate. The models selected by the AIC, BIC, adjR^2 and F-test may need further judgements in deciding on the final model which can take two forms. First, choosing a model half way through the search process just before it chooses a more complex model with worse performance on the test data. Second, pruning out some of the variables with small coefficients. The need for these manual adjustments explains

the real reason behind the traditional common practice of specifying beforehand the maximum number of variables for a model (e.g.[10], [7]).

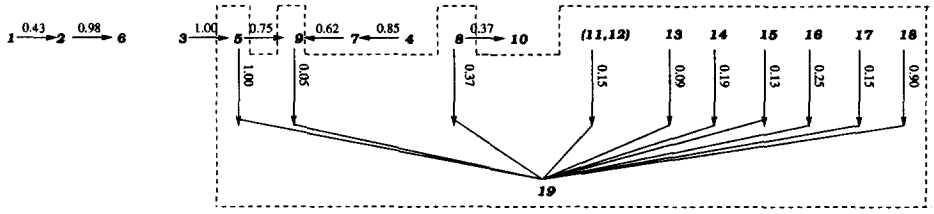


Fig. 1. Model 1. The independent variables and the link coefficients to be discovered are in the dashed-line box, i.e. only variables with direct links to the target variable

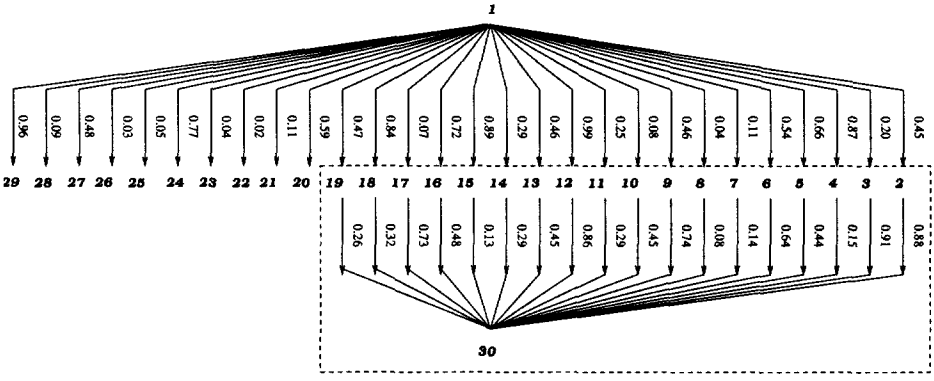


Fig. 2. Model 2. Variable 1 is directly linked to all of the variables with direct links (some of which are very weak) to the target variable. Large link coefficients are deliberately placed between variable 1 and these variables to see if this will cause variable 1 to also be chosen

4 Conclusion

A version of Minimum Message Length model selection method and a non-backtracking search strategy have been developed. Using the common search strategy, the robustness of a number of model selection methods in performing the task of selecting models that balance model complexity and goodness of fit is examined.

Based on the experiments with artificial data and real atmospheric data for hurricane intensity change forecasting, it has been shown that MML, MDL, CAICF, SRM and SC methods are good candidates for fully automated model

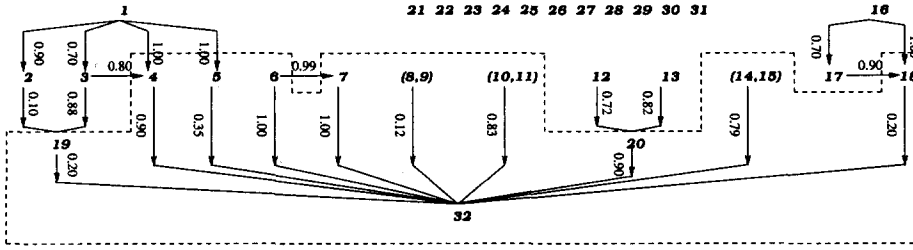


Fig. 3. Model 3. Random values with unit normal with no link to the target variable are generated for variables 21 to 31 are included in the pool of potential variables

Table 2. Performance of the different model selection methods on the task of discovering Model 1, 2 and 3 using varying sample sizes

Sample	Method	Model 1 (nvar= 10)				Model 2 (nvar= 18)				Model 3 (nvar= 10)			
Size		nvar	ModelErr	RMSE	R^2	nvar	ModelErr	RMSE	R^2	nvar	ModelErr	RMSE	R^2
500	MML	6	0.0109	1.0399	0.7285	14	0.0068	1.0563	0.9453	10	0.0027	1.0245	0.9266
	MDL	6	0.0109	1.0399	0.7285	14	0.0068	1.0563	0.9453	10	0.0027	1.0245	0.9266
	CAICF	6	0.0109	1.0399	0.7285	14	0.0170	1.0762	0.9432	11	0.0050	1.0475	0.9234
	SRM	6	0.0109	1.0399	0.7285	17	0.0048	1.0635	0.9449	17	0.0108	1.1242	0.9129
	SC	8	0.0045	1.0320	0.7337	22	0.0077	1.1264	0.9388	20	0.0120	1.1624	0.9074
	AIC	40	0.0093	1.1858	0.6713	70	0.0130	1.4311	0.9112	70	0.0145	1.5098	0.8601
	BIC	40	0.0093	1.1858	0.6713	70	0.0130	1.4311	0.9112	70	0.0145	1.5098	0.8601
	adjR^2	70	0.0093	1.3232	0.6174	70	0.0130	1.4311	0.9112	70	0.0143	1.5126	0.8596
	F-test	15	0.0052	1.0523	0.7270	43	0.0130	1.3027	0.9218	30	0.0119	1.2276	0.8989
1000	MML	8	0.0049	1.0367	0.7567	17	0.0012	0.9894	0.9539	10	0.0008	1.0453	0.9256
	MDL	8	0.0049	1.0367	0.7567	17	0.0012	0.9894	0.9539	10	0.0008	1.0453	0.9256
	CAICF	8	0.0049	1.0367	0.7567	17	0.0012	0.9894	0.9539	10	0.0008	1.0453	0.9256
	SRM	8	0.0049	1.0367	0.7567	17	0.0012	0.9894	0.9539	13	0.0035	1.0722	0.9220
	SC	10	0.0057	1.0427	0.7544	23	0.0024	1.0123	0.9521	16	0.0049	1.0875	0.9200
	AIC	30	0.0035	1.0684	0.7473	70	0.0059	1.1577	0.9403	70	0.0062	1.2322	0.9029
	BIC	30	0.0035	1.0684	0.7473	70	0.0059	1.1577	0.9403	70	0.0062	1.2322	0.9029
	adjR^2	62	0.0061	1.1286	0.7273	70	0.0058	1.1656	0.9395	70	0.0062	1.2322	0.9029
	F-test	17	0.0043	1.0395	0.7576	42	0.0070	1.1128	0.9432	40	0.0056	1.1686	0.9099
2000	MML	9	0.0014	1.0103	0.7636	17	0.0013	1.0011	0.9509	9	0.0022	0.9945	0.9307
	MDL	7	0.0047	1.0224	0.7577	17	0.0013	1.0011	0.9509	10	0.0006	0.9857	0.9319
	CAICF	7	0.0047	1.0224	0.7577	17	0.0013	1.0011	0.9509	10	0.0006	0.9857	0.9319
	SRM	7	0.0047	1.0224	0.7577	17	0.0013	1.0011	0.9509	10	0.0006	0.9857	0.9319
	SC	9	0.0014	1.0103	0.7636	20	0.0011	1.0090	0.9502	19	0.0030	1.0025	0.9299
	AIC	31	0.0014	1.0316	0.7562	70	0.0027	1.0767	0.9447	70	0.0025	1.0745	0.9216
	BIC	31	0.0014	1.0316	0.7562	70	0.0027	1.0767	0.9447	70	0.0025	1.0745	0.9216
	adjR^2	65	0.0016	1.0597	0.7473	70	0.0025	1.0787	0.9445	70	0.0025	1.0745	0.9216
	F-test	13	0.0011	1.0131	0.7628	40	0.0022	1.0443	0.9472	37	0.0028	1.0265	0.9272
4000	MML	9	0.0005	1.0013	0.7743	18	0.0004	0.9885	0.9535	10	0.0002	0.9899	0.9316
	MDL	9	0.0005	1.0013	0.7743	18	0.0004	0.9885	0.9535	10	0.0002	0.9899	0.9316
	CAICF	9	0.0005	1.0013	0.7743	18	0.0004	0.9885	0.9535	10	0.0002	0.9899	0.9316
	SRM	9	0.0005	1.0013	0.7743	18	0.0004	0.9885	0.9535	10	0.0002	0.9899	0.9316
	SC	10	0.0001	0.9994	0.7753	22	0.0009	0.9955	0.9529	12	0.0005	0.9907	0.9315
	AIC	38	0.0011	1.0151	0.7697	70	0.0013	1.0221	0.9509	70	0.0011	1.0287	0.9272
	BIC	38	0.0011	1.0151	0.7697	70	0.0013	1.0221	0.9509	70	0.0011	1.0287	0.9272
	adjR^2	56	0.0009	1.0210	0.7681	70	0.0013	1.0221	0.9509	70	0.0011	1.0287	0.9272
	F-test	18	0.0008	1.0061	0.7727	46	0.0012	1.0119	0.9516	30	0.0011	1.0065	0.9296
6000	MML	10	0.0002	1.0106	0.7697	18	0.0002	1.0018	0.9518	10	0.0006	1.0055	0.9300
	MDL	10	0.0002	1.0106	0.7697	18	0.0002	1.0018	0.9518	10	0.0006	1.0055	0.9300
	CAICF	10	0.0002	1.0106	0.7697	18	0.0002	1.0018	0.9518	10	0.0006	1.0055	0.9300
	SRM	9	0.0006	1.0131	0.7686	18	0.0002	1.0018	0.9518	10	0.0006	1.0055	0.9300
	SC	10	0.0002	1.0106	0.7697	19	0.0002	1.0026	0.9517	10	0.0006	1.0055	0.9300
	AIC	32	0.0006	1.0182	0.7672	68	0.0007	1.0245	0.9500	70	0.0008	1.0318	0.9270
	BIC	*32*	*0.0006*	*1.0182*	*0.7672*	*68*	*0.0007*	*1.0245*	*0.9500*	*70*	*0.0008*	*1.0318*	*0.9270*
	adjR^2	62	0.0005	1.0259	0.7648	70	0.0007	1.0240	0.9500	70	0.0008	1.0318	0.9270
	F-test	14	0.0004	1.0130	0.7688	30	0.0005	1.0113	0.9509	29	0.0012	1.0180	0.9285
10000	MML	10	0.0001	1.0116	0.7702	18	0.0001	1.0014	0.9513	10	0.0001	1.0207	0.9290
	MDL	10	0.0001	1.0116	0.7702	18	0.0001	1.0014	0.9513	10	0.0001	1.0207	0.9290
	CAICF	10	0.0001	1.0116	0.7702	18	0.0001	1.0014	0.9513	10	0.0001	1.0207	0.9290
	SRM	10	0.0001	1.0116	0.7702	18	0.0001	1.0014	0.9513	10	0.0001	1.0207	0.9290
	SC	10	0.0001	1.0116	0.7702	20	0.0002	1.0024	0.9512	11	0.0002	1.0211	0.9290
	AIC	32	0.0005	1.0162	0.7686	70	0.0005	1.0159	0.9502	70	0.0005	1.0349	0.9275
	BIC	32	0.0005	1.0162	0.7686	70	0.0005	1.0159	0.9502	70	0.0005	1.0349	0.9275
	adjR^2	52	0.0004	1.0193	0.7677	70	0.0005	1.0159	0.9502	70	0.0004	1.0343	0.9276
	F-test	13	0.0003	1.0128	0.7697	43	0.0005	1.0095	0.9507	22	0.0004	1.0240	0.9286

selection tasks. Given a noisy data set, the methods can reliably converge to a true model (if one exists) or to a reasonably parsimonious model.

The fact that AIC, BIC, adjR^2 and F-test have overfitted the training data suggests that when comparing two models with different complexity, the increase in the penalty terms for model complexity is not sufficient compared to the decrease in the terms for goodness of fit. This prompted the doubt that the balancing mechanism of the methods might not be robust enough for automated model selection task.

Acknowledgments

The author is grateful to Chris Wallace for guidance in the development of the MML method. The author is a recipient of the Australian Postgraduate Award (Industry).

References

1. H. Akaike. Information theory and an extension of the Maximum Likelihood principle. In B.N. Petrov and F. Csaki, editors, *Proc. 2nd Int. Symp. Information Thy.*, pages 267-281, 1973.
2. R. Baxter and D. Dowe. Model selection in linear regression using the MML criterion. Technical Report 276, School of Computer Science and Software Engineering, Monash University, 1996.
3. H. Bozdogan. Model selection and akaike's information criterion (AIC): the general theory and its analytical extensions. *Psychometrika*, 52(3):345-370, 1987.
4. J.H. Conway and N.J.A. Sloane. *Sphere Packings*, Lattices and Groups. Springer-Verlag, New York, 1988.
5. M. Ezekiel. *Methods of Correlation Analysis.* Wiley, New York, 1930.
6. C. Mallows. Some comments on C_p. Technometrics, 15:661-675, 1973.
7. A.J. Miller. *Subset Selection in Regression.* Chapman and Hall, London, 1990.
8. J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465-471, 1978.
9. J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society B*, 49(1):223-239, 1987.
10. T. Ryan. *Modern Regression Methods.* John Wiley & Sons, New York, 1997.
11. G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461-464, 1978.
12. V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.
13. C.S. Wallace. On the selection of the order of a polynomial model. unpublished technical report, Royal Holloway College, 1997.
14. C.S. Wallace and P.R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society B*, 49(1):240-252, 1987.

Frequent Itemset Counting Across Multiple Tables*

Viviane Crestana Jensen and Nandit Soparkar

Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109-2122
{viviane,soparkar}@eecs.umich.edu

Abstract. Available technology for mining data usually applies to centrally stored data (i.e., homogeneous, and in one single repository and schema). The few extensions to mining algorithms for decentralized data have largely been for load balancing. In this paper, we examine mining decentralized data for the task of finding frequent itemsets. In contrast to current techniques where data is first joined to form a single table, we exploit the inter-table foreign key relationships to obtain decentralized algorithms that execute concurrently on the separate tables, and thereafter, merge the results. In particular, for typical warehouse schema designs, our approach adapts standard algorithms, and works efficiently. We provide analyses and empirical validation for important cases to exhibit how our approach performs well. In doing so, we also compare two of our approaches in merging results from individual tables, and thereby, we exhibit certain memory vs I/O trade-offs that are inherent in merging of decentralized partial results.

1 Introduction

Data mining (DM) algorithms, a part of *knowledge discovery in databases*, are typically designed for centralized data (i.e., homogeneous data stored in one central repository, with a central administration, and in a single table). However, information may be dispersed among different tables, and in some cases, the tables may reside in different physical locations. Our research, as presented in this paper, describes and analyzes our DM algorithms for decentralized tables.

DM itself is generally performed on data stored in data warehouses, and even so, the data may not be stored in a single table as is assumed by most available algorithms (e.g., see [3,14,9]). For instance, the star schema [12] used in data warehouses is organized into two groups: facts (the core data being analyzed) and dimensions (the attributes about the facts); the fact table is usually much larger than the dimension tables. If typical *association rules* (AR) algorithms (e.g., see [3]) were applied to data stored in a star schema, it would be necessary to compute first the join of the fact table with its dimension tables. Even though

* This work was initiated when the authors were visiting IBM T.J. Watson Research Center, and was supported partially by IBM Research funds.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 49–61, 2000.
© Springer-Verlag Berlin Heidelberg 2000

the cost of a join is often overshadowed by that of running DM, a joined table has many more columns and rows (which normalization had been used to avoid), and this affects adversely the cost of the DM algorithms as well.

We begin with a concrete example: star schema tables which represent a case of tables associated by foreign key relationships. Motivated by pragmatic considerations stated above, we present an algorithm that adapts and extends available algorithms to work efficiently with such decentralized tables. We exploit the decentralization by executing the algorithms concurrently on the separate tables, and thereafter, we merge the results. This approach requires modifying available algorithms, and considering new performance factors. We present our analytical and empirical evaluation for particular cases to illustrate our performance gain.

2 Background and Related Work

We review the problem of discovering AR using relevant related work, and illustrate the problems in applying traditional techniques to decentralized data.

2.1 Related Work

The AR discovery problem may be described as follows. Given a set of items I, and a set of records T (i.e., a table), where each record t_j is composed of a subset i_j of the set of items I, the problem is to find associations among the items such that the presence of some items in a record will suggest the presence of other items in the same record. An AR, denoted by $X \Rightarrow Y$, where X and Y are subsets of a set of items I, is said to have a *confidence* of c, $c \in [0,1]$, iff $(100 * c)\%$ of the records in the database which contain the items in X also contain the items in Y. The *support* for such a rule is defined to be the fraction of records in the table which contain the items in $X \cup Y$. The problem is to find all rules that meet a user-specified minimum confidence and support.

Discovering AR as introduced in [2], is improved in [3] by the *Apriori* algorithm in terms of performance. The problem is decomposed into: (1) finding the large (i.e., frequent) itemsets (i.e., which meet a user-defined support threshold); and (2) generating the AR, based on the support counts found in step 1. The research in AR concentrates on the performance of the expensive first step (e.g., see [10,11,4,1]), as is the case in this paper. The Apriori algorithm performs the counting of itemsets in an iterative manner, by counting the itemsets that contain k items (k-itemsets) at iteration k. In each iteration, a candidate set of frequent itemsets is constructed, and the table is scanned to count the number of occurrences of each candidate itemset.

AR were initially studied in the context of market basket data (see [2]), and new types of AR have been considered as well (e.g., [14]). Other algorithms based on [3] have been presented (e.g., [10,11,4,1]) – but they all assume that the data is stored in one single table. Other work on distributing the Apriori algorithm (e.g., [5]) considers databases that are horizontally, but not vertically, partitioned. That is, one horizontally partitioned table is assumed, with the

partitions stored at different sites; each site has the same schema. A minor variation of the Apriori algorithm is run at the different sites, and the mined results from the different partitions are merged. However, the different sites have to synchronize after each pass, and the amount of data processed is essentially the same as in a sequential algorithm (other than for message exchanges) – i.e., only the load is shared. In contrast, we provide an approach that applies to vertically partitioned tables, and thereby, the cost of processing is reduced.

2.2 Decentralized Tables

We illustrate some problems in discovering AR for decentralized data as an example. The following is a schema from a banking environment.

- $Customer(\underline{acct\#}, name, balance, zipcode, age)$
- $ATM(\underline{atm\#}, type, street, zipcode, limit)$
- $ATMactivity(\underline{xact\#}, acct\#, atm\#, amount)$

In Figure 1, we show a relevant projection of the tables,[1] assuming that the quantitative attributes (e.g., age and monetary amounts) are partitioned using an appropriate algorithm (e.g., see [14]).

When accesses are limited to single tables, the traditional approaches to discovering AR would work well for finding associations such as:

- $< age : 20..29 > \Rightarrow < balance : 1000..1999 >$ for table *Customer*
- $< type = in > \Rightarrow < limit : 10000..19999 >$ for table *ATM*

However, the same approaches will not work well to find:

- $< type = drive > \Rightarrow < age : 20..29 >$

For rules involving more than one table, we would first need to join the tables (i.e., $Customer \bowtie ATMactivity \bowtie ATM$). The significant redundancy in such a joined table would degrade performance (e.g., the itemset $< age : 30..39 > AND < area : x >$ occurs three times in the final table, and therefore, would need to be counted three times by the algorithm; and yet, it corresponds to just one entry in the customer table for primary key $acct\# = 05$).

3 A Decentralized Approach

We suggest a two-phase counting strategy for the frequent itemset discovery:

- Find frequent itemsets on individual tables separately; and then
- Merge results from individual tables by using the foreign key relationships.

To begin with, we use the example from Section 2 to illustrate our approach.

[1] The primary key for the each table is underlined, and the repeat entries in $ATMactivity$ for $acct\#$ and $atm\#$ correspond to different $xact\#$'s.

Customer Table			
acct#	balance	zipcode	age
01	1000..1999	x	20..29
02	1000..1999	z	20..29
03	1000..1999	y	20..29
04	2000..5000	y	30..39
05	2000..5000	x	30..39
06	1000..1999	z	30..39

ATM Table			
atm#	type	zipcode	limit
A	drive	x	0..9999
B	out	y	0..9999
C	out	z	0..9999
D	in	x	10000..19999
E	in	y	10000..19999
F	in	z	10000..19999

ATMactivity Table		
acct#	atm#	amount
01	A	15..25
01	A	15..25
02	A	15..25
02	C	50..100
02	C	50..100
03	A	15..25
03	B	15..25
04	B	50..100
04	E	500..1000
05	A	15..25
05	A	15..25
05	D	50..100
06	C	50..100
06	F	500..1000

Fig. 1. Relevant projection of the three tables.

3.1 Illustrative Algorithm

In the first phase, we count itemsets on the individual tables of *Customer* and *ATM* separately. The itemsets from each individual table should be frequent with respect to the final joined table – e.g., in the example of Section 2.2, if we consider support on the individual tables alone, the itemset $< type : drive, zipcode : x >$ would have support of 0.167 for the table *ATM*, whereas for the joined table it should be 0.429. The reason for the different support values is due to the number of occurrences of the record $atm\# = A$ in the *ATMactivity* table. By determining the number of occurrences of a given record as it would be in the final joined table, (which, in this case, happens to be the number of occurrences in the *ATMactivity* table), we can modify the Apriori algorithm (for instance) in order that, when counting itemsets, this correct number of occurrences is taken into account.

We also need to count itemsets whose items span more than one table. Now, if an itemset is frequent, all of its subsets are also frequent [3]. Therefore, all frequent itemset's subsets, such that all items come from any one table, are frequent, and these subsets would have been found frequent when our algorithm ran at that particular table. As a result, for cross table frequent itemsets in our example, we only need to consider itemsets that are a concatenation of two frequent itemsets: one from table *Customer* and one from table *ATM*.[2]

Let table $T = Customer \bowtie ATMactivity \bowtie ATM$. Let an itemset from table T be $I_T = I_{Customer} \cup I_{ATM}$, where $I_{Customer}$ and I_{ATM} contain items

[2] For brevity, we disregard *amount* in the *ATMactivity* table; [6] describes handling this.

that belong to tables *Customer* and ATM, respectively. In order for an itemset I_T to be present in table T, a record $r_{Customer}$ (that contained $I_{Customer}$) from table *Customer* must be present together in table T with a record r_{ATM} (that contained I_{ATM}) from table ATM. That is, table $ATMactivity$ must have an entry corresponding to $(\pi_{acct\#}(r_{Customer}), \pi_{atm\#}(r_{ATM}))$. To count I_T, we use a 2-dimensional array where the elements in each dimension correspond to the itemsets found frequent in each of the tables *Customer* and ATM. Let $l_{Customer}$ and l_{ATM} be the sets of frequent itemsets present in tables *Customer* and ATM, respectively. By examining each entry in the $ATMactivity$ table, we determine the records in the two tables to be joined to form a record in table T. By considering each pair of frequent itemsets, one from $l_{Customer}$ and one from l_{ATM}, we determine which itemsets are present in table T for a corresponding entry in the $ATMactivity$ table. Therefore, by using the 2-dimensional array and one scan of table $ATMactivity$, we count all frequent itemsets in table T.

3.2 Algorithm for Star Schema

Let a *primary table* be a relational table with one primary key and no foreign keys; the table may have other fields that are categorical, numerical or boolean. Also, let a *relationship table* be a relational table which contains foreign keys to other *primary tables*. Typically, primary tables refer to entities, and relationship tables correspond to relationships in an ER diagram ([13]).

We present a decentralized version of the Apriori algorithm, as an example case, for star schemas in which there are n primary tables (the *dimension tables*): $T_1, T_2, \cdots, T_n$; and one central relationship table (the *fact table*): T_{1n}. Each $T_t(id_t, a_{t1}, \cdots, a_{tm_t})$ has id_t as primary key, and $T_{1n}(id_1, id_2, \cdots, id_n)$ has id_t as foreign key to table T_t. Our algorithm finds frequent itemsets on table $T = T_{1n} \bowtie T_1 \bowtie T_2 \bowtie \cdots \bowtie T_n$ as follows.

Phase I: *Count itemsets on primary tables.*

1. *Compute a projection of the relationship table*: Access relationship table T_{1n} to count the number of occurrences of each value for the foreign keys id_t. Store the number of occurrences in a vector v_t for each table T_t. Note: the number of elements in v_t equals the number of rows in T_t.

2. *Counting frequent itemsets on individual primary tables*: Count itemsets using the Apriori algorithm as in [3] on the primary tables, but modified as follows: in primary table T_t, when incrementing the support of an itemset present in row i by 1, increment the value of the support by the number of occurrences of the value of id_t for row i in table T_{1n} (i.e., the ith element of v_t). We refer to this step of our algorithm as the "modified Apriori" in the remainder of this paper. In this manner, we find the n sets of frequent itemsets, l_t from table T_t; the itemsets are of length 1 up to m_t (the number of attributes in table T_t).

Phase II: *Count itemsets across primary tables using the relationship table.* Generate candidates from the n primary tables using an n-dimensional count array, where dimension t corresponds to elements of the set l_t together with the

empty set.[3] We compute the joined table T without materializing it, processing each row as it is formed, thereby avoiding storage costs for T. Now, for each row r in T, consider the attributes in $\pi_{T_t}(r)$ that come from each table T_t, and identify the subset of itemsets from l_t contained in $\pi_{T_t}(r)$, say i_t. After computing all such i_t, $t = 1..n$, increment by one each position in the n-dimensional array whenever an element I_T is formed by concatenating an element of i_1, an element of i_2, ..., and an element of i_n (i.e., $i_1 \times i_2 \times \cdots \times i_n$). That is, I_T exemplifies an itemset contained in table T, such that its items belong to more than one primary table. After table T is processed in this manner, the n-dimensional array contains the support for all the candidate itemsets, and we can use it to determine which itemsets are indeed frequent.

Our algorithm, running on decentralized tables, finds the same frequent itemsets that the Apriori algorithm does running on the joined table. Omitting a proof here due to space constraints, we indicate that it is a formalization of the arguments presented in Section 3.1. Note that though our approach uses the Apriori algorithm to perform counting in the primary tables, most other centralized algorithms could be modified to use the occurrence vector v_t. We could choose faster centralized algorithms (e.g., [10,11,4,1,7]), and thereafter, merge the partial results by using our techniques. Also, although we illustrated decentralized DM for the case of the star schema, our basic approach may be extended to more general designs, and [8] provides an approach to do so. An immediate advantage of our two-phase approach is that table T is computed and processed only once, and therefore, there is no need to store it. This comes at the expense of possibly generating more cross-table itemsets as compared to Apriori, and we examine this issue below.

3.3 Re-examining the Merging

A major consideration in our decentralized algorithms is in the merging phase; the counting of itemsets at the individual tables is relatively simple. Our algorithm of Section 3.2, which we call the **I/O saving** approach, has I/O costs saved using multiple scans on smaller tables (as compared to scans over the larger table T). Also, we save on processing time since a given frequent itemset consisting of items from only one primary table will be counted fewer times than if we counted on the table T (see Section 4). However, for itemsets with items from more than one table, there is no pruning from one pass to the next because all itemsets are counted in one scan during the merge. Therefore, we may end-up considering some additional candidates (which we call "false candidates") than if we were to perform pruning at every pass.

If the sets l_t are too large, our approach may require considerable memory space; however, we do save on some costs since all the itemsets belonging to an individual table are already considered at the merge point, and as a result, we effectively prune away many potential cross-table itemsets. In any case, if the n-dimensional array does not fit in memory, we may resort to a **Memory saving**

[3] The empty set allows for candidates from more than one, but less than n, tables.

merge algorithm for Phase II. This different approach does build the join of the $n+1$ tables, creating table T – but only at the merge phase. The counting step of the original Apriori algorithm is run on table T, but generates only the cross table candidate itemsets. The pruning step uses the frequent itemsets l_t, as well as the cross table itemsets generated. In the Memory saving merge the I/O costs could be higher since we may have to scan table T several times. However, we avoid counting false candidates.

There is potential to use a hybrid I/O and Memory saving merge approach as follows. We can build table T, and perform the Memory saving approach by scanning it a few times until the remaining subsets of l_t fit in memory. Thereafter, a switch may be made to the I/O saving approach. A limited version of this algorithm proved to work well in experiments (see Section 5). Alternatively, before merging all the results from Phase I, we can process the primary tables pairwise (e.g., $T_1 \bowtie T_{1n} \bowtie T_2$), using the I/O saving merge by considering the sets l_1 and l_2 as candidates. In this way, we increase the pruning when considering, say, T_1, T_2 and T_3, since we know the false candidates that involve attributes of T_1 and T_2 earlier. Also, we can consider table $T_1 \bowtie T_{1n} \bowtie T_2$ in place of T_1 and T_2, consequently reducing the number of dimensions. Here, we would not need to materialize table T, but the number of joins that we would compute (first pair-wise, then all tables) are more.

Another aspect of the merging process is the handling of categorical attributes in the relationship table; [6] offers a discussion.

4 A Cost Analysis

For the case of the star schema, we compare the costs of running the original Apriori algorithm on the joined table with the costs of running the algorithms that we propose. We consider both options for Phase II (i.e., the I/O saving and Memory saving approaches); and the processing costs considered are for I/O and computing, and we examine each after providing some nomenclature. For details of the derivations, please see [6].

4.1 Nomenclature

Assume n primary tables $T_t, t = 1..n$ and a relationship table T_{1n} as described in Section 3.2; and assume that the primary tables are ordered in their primary keys. Furthermore, the tables T_t have r_t records, and table T_{1n} has r_{1n} records. We assume that $r_t \ll r_{1n}, \forall t$. To run the Apriori frequent itemset counting algorithm, the join of the $n+1$ tables, $T = T_{1n} \bowtie T_1 \bowtie T_2 \bowtie \ldots \bowtie T_n$, is materialized, and T has r_{1n} records with attributes $(id_1, a_{11}, \cdots, a_{1m_1}, id_2, a_{21}, \cdots, a_{2m_2}, \cdots, id_n, a_{n1}, \cdots, a_{nm_n})$.

Let, k be the length of the longest candidate itemset on T; $|c_j|$ be the number of candidates of length j on T; k_t be the length of the longest candidate itemset on T_t; k_{1n} be the length of the longest candidate itemset on T, such that the items belong to more than one primary table ($k_{1n} \leq k$); $|c_j^t|$ be the number of

candidates of length j where all the items belong to table T_t, $t = 1..n$; $|c_j^{1n}|$ be the number of candidates of length j where the items belong to more than one primary table; and $|l_j^t|$ be the number of frequent itemsets of length j where all the items belong to table T_t.

4.2 I/O Costs

I/O costs are directly related to the table size and the number of scans; the cost of accessing from disk a single atomic value in a table is the unit for I/O costs.

1. Apriori on table T:

$$cj + r_{1n} \sum_{t=1}^{n}(m_t) + k * r_{1n} \sum_{t=1}^{n}(m_t)$$

where the first term is to compute the join (cj) of the tables (cj represents the I/O cost of whichever join algorithm is used), the second term is for writing out the computed join, and the last term is for table scans when the Apriori algorithm is run (k scans of the table).

2. Decentralized with I/O saving merge:

$$n * r_{1n} + \sum_{t=1}^{n} r_t + \sum_{t=1}^{n} k_t(m_t + 1)r_t + cj$$

where the first two terms correspond to step 1 of Phase I: first counting occurrences, and then storing the occurrences to be used in the next step. The third term corresponds to step 2 of Phase I (i.e., running the modified Apriori algorithm on the primary tables), and the last term corresponds to Phase II: computing the join, but without saving the result of the join.

3. Decentralized with Memory saving merge:

$$n * r_{1n} + \sum_{t=1}^{n} r_t + \sum_{t=1}^{n} k_t(m_t + 1)r_t + cj + r_{1n} \sum_{t=1}^{n}(m_t) + k_{1n} * r_{1n} \sum_{t=1}^{n}(m_t)$$

when compared to I/O saving, we add the costs of saving the joined table and scanning k_{1n} times the table T.

For the I/O saving approach, we see that the dominant term is the scanning of the primary tables (k_t times for each table T_t), for the Memory saving approach, the dominant term is scanning k_{1n} times the table T, and for the Apriori on table T approach, the dominant term is k scans of table T. Given that $k_t \leq k$, and $r_t \ll r_{1n}$, we see that our first approach offers I/O cost savings as compared to first computing the join and then running the Apriori algorithm. For the Memory saving approach, savings in I/O costs are not assured; they depend on how k_{1n} compares to k. In practice, k_{1n} could be much smaller than k since related items are often clustered.

4.3 Compute Costs

For simplicity, we only discuss the dominant CPU cost: given a row and a set of candidate itemsets, determining the candidates present in the row (called the *subset function*). See [6] for a more complete cost model.

For every scan, the subset function is executed for each row of the table. It is difficult to assess the cost for this function; generally, the cost increases with the length of the row and the number of candidates. With $f(p,q)$ as the cost of the subset function for a row of length p and a candidate set of size q, CPU compute costs are:

1. Apriori on table T:

$$\sum_{j=1}^{k} r_{1n} * f(m, |c_j|)$$

for each iteration j, with m as the length of a row in T.

2. Decentralized with I/O saving merge:

$$\sum_{t=1}^{n} \sum_{j=1}^{k_t} r_t * f(m_t, |c_j^t|) + \sum_{j=1}^{k} r_{1n} * f(m_t, |l_j^t|)$$

where the first term arises from the modified Apriori at the primary tables, and the second term from Phase II.

3. Decentralized with Memory saving merge:

$$\sum_{t=1}^{n} \sum_{j=1}^{k_t} r_t * f(m_t, |c_j^t|) + \sum_{j=1}^{k_{1n}} r_{1n} * f(m, |c_j^{1n}|)$$

where $m = \sum_{t=1}^{n} m_t$ and $|c_j^{1n}| = |c_j| - \sum_{t=1}^{n} |c_j^t|$. The first term is as above, and the second term is for counting cross table itemsets.

The initial terms in our I/O saving approach are much smaller than corresponding ones for Apriori on table T since $m_t < m$, $|c_j^t| < |c_j|$, $k_t < k$ and $r_t \ll r_{1n}, \forall t$. For the terms in the second summation, it is the case that $|l_j| \leq |c_j|$, and in many cases, $|l_j| \ll |c_j|$. In fact, the multiplicand could be less than r_{1n} depending on the statistical distribution of the data values (e.g., in case two entries in T_{1n} with the same value for id_1 happen to be close enough so that the subset function does not need to be recomputed) – which cannot be exploited by the Apriori on table T. For our Memory saving merge, we notice that the only term multiplied by r_{1n} has a considerably smaller set of candidates in our approach than in the original Apriori.

5 Empirical Validation

We restricted our attention to the case with three tables: two primary tables and one relationship table; our analysis indicates that our approach is likely to perform better with more involved database designs where the final joined table is much larger than the sum of the sizes of the decentralized tables.

5.1 Experimental Setup

We ran experiments on synthetic data using the data generator in [3]. Since our study is for decentralized data, and [3] produces centralized data (i.e., one table), we used the generator for the primary tables T_1 and T_2 with parameters: N, the

number of items; $|D|$, the number of records; $|T|$, the average length of records; and $|I|$ the average length of the maximal potentially frequent itemsets.

To generate T_{12}, we used our own generator with parameters: $|D|$, the number of records in the final table $T = T_1 \bowtie T_{12} \bowtie T_2$; and R, the average number of repetitions of entries in T_{12} (note: $r_{12} \leq r$). Each line generated for table T_{12} has randomly picked records from tables T_1 and T_2, and the number of repetitions selected from a Poisson distribution (with mean = R). The join of the three tables, T, is generated in order to compare with the original Apriori. After computing the join, we determined the average length of records for table T. The cost of the join was not included in the results, since all the algorithms have to compute the join at some point (without necessarily materializing it).

We implemented the I/O saving, the Memory saving, and limited Hybrid (in which we switch to the I/O saving approach after the first pass) approaches. Our experiments used a 200 MHz Pentium Pro machine with a 256 Mbytes RAM, running Linux. Among our extensive evaluations, two representative experiments have their parameters listed in Figure 2.

Parameters for testing											
	T_1				T_2				T_{12}		T
Test	N	$\|D\|$	$\|T\|$	$\|I\|$	N	$\|D\|$	$\|T\|$	$\|I\|$	$\|D\|$	R	$\|T\|$
Test 1	0.5K	1K	10	4	0.55K	1.2K	10	4	0.1M	50	20
Test 2	0.5K	0.1M	6	4	0.55K	0.1M	7	4	10M	300	13

Fig. 2. Parameters for testing.

We used various support values with the hash tree for the Apriori (see [3]) always fitting in memory. If the hash tree were to not fit in memory, our approach would be even better, given that at each pass, the number of candidates that our approach examines is smaller than the original Apriori.

5.2 Experimental Findings

We present our results as follows: first, we plot the time taken by the Apriori algorithm (i.e., our base case) when evaluated on table T, and second, the time taken by our approach divided by the time taken by the base case approach, referred to as our "normalized" results; results are presented in Figures 3 and 4.

For Test 1, the files were small (Table T with only 100000 records), and our goal was to verify whether our approach could provide some CPU savings, considering that the entire table fit in main memory. We verified that our approach performed better, with the exception of the I/O saving algorithm. The reason for the limited performance of the I/O saving approach is that, as explained in Section 3.2, when there are many candidates from the primary tables, the number of false candidates grows significantly. For lower values of support, the savings provided are not so significant, but our algorithm still performs comparable to Apriori. For Test 2, the files were larger, and the savings accrued due to

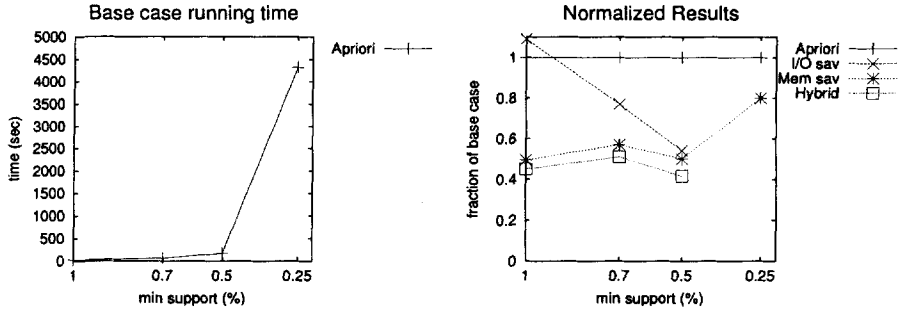


Fig. 3. Results for Test 1.

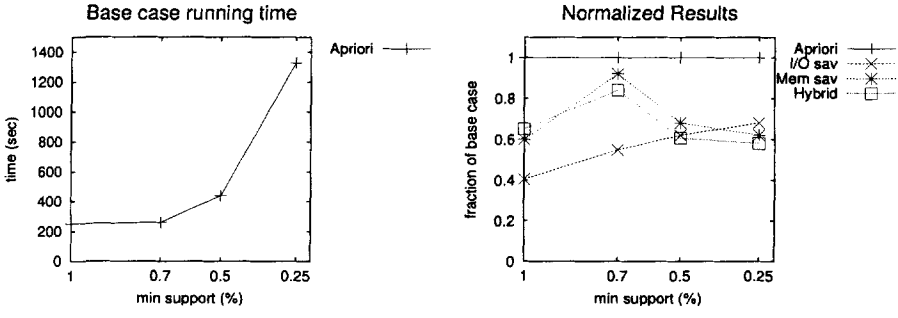


Fig. 4. Results for Test 2.

fewer scans of the big table T. We see that the I/O saving algorithm performs well for all support values, an advantage with our approach since the I/O saving algorithm need never materialize the computed join.

Our algorithms were run serially; first we ran modified Apriori on table T_1, then we ran modified Apriori on table T_2, and then Phase II of our algorithm on table T to compute frequent itemsets across tables. Therefore, the times shown account for the running time on the primary tables as well. If the processing were concurrently on the original tables, the make-span of the running times for our approach would be even lower due to parallelism.

5.3 Comparisons with Cost Analysis

To compare the empirical results with our cost model, our code kept track of the subset function costs; a counter was incremented each time a node in the hash tree was accessed, and each time an itemset contained in a leaf was checked against a record. Similarly, we monitored the actual time spent in I/O. The results obtained in comparing with our modeling, showed that our cost analysis is reasonably accurate; see [6] for details.

6 Conclusions

In this paper, we examined the issues in mining of data stored in decentralized tables. We described counting of frequent itemsets (used for association rules discovery), without requiring to materialize a join of the decentralized tables. As a basic case, we applied our approach to a star schema in which several smaller dimension tables are associated by foreign keys to a central fact table, and we presented efficient algorithms that adapts some available approaches. Previous approaches required the separate tables to be joined to form a single table before the data could be mined. In contrast, we exploited foreign key relationships to develop decentralized algorithms that execute concurrently on the separate tables, and thereafter, we merge the results. We examined the important issues in merging partial results, the compute and memory requirements, and the trade-offs that arise. Furthermore, we provided analyses to assess the performance of our techniques, and we presented empirical validation. Our research indicates the issues and feasibility of mining of decentralized datasets which will become increasingly important as focus shifts toward scaling-up to real-life datasets.

Acknowledgements

The authors thank Ramesh C. Agarwal and Anant Jhingran of IBM Research for valuable suggestions in the early stages of this work.

References

1. R. C. Agarwal, C. C. Aggarwal, V. V. V. Prasad, and V. Crestana. A tree projection algorithm for generation of large itemsets for association rules. IBM Research Report: RJ 21246. IBM Research Division, New York, 1998.
2. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of ACM-SIGMOD Int'l Conference on Management of Data*, 1993.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of 20th Int'l Conference on Very Large Data Bases*, 1994.
4. S. Brin, R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. of ACM-SIGMOD Int'l Conference on Management of Data*, 1997.
5. D. Cheung, V. Ng, A. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. *IEEE Transactions on Knowledge & Data Engineering*, 1996.
6. V. Crestana and N. Soparkar. Mining decentralized data repositories. Tech Report: CSE-TR-385-99. The University of Michigan, Ann Arbor. February 1999.
7. B. Dunkel and N. Soparkar. Data organization and access for efficient data mining. In *Proc. of 15th IEEE Int'l Conference on Data Engineering*, 1999.
8. V. C. Jensen and N. Soparkar. Algebra-based optimization strategies for decentralized mining. Tech Report: CSE-TR-418-99. The University of Michigan, Ann Arbor. December 1999.

9. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of 4th Int'l Conference on Knowledge Discovery & Data Mining*, 1998.
10. J. S. Park, M-S Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. of ACM-SIGMOD Int'l Conference on Management of Data*, 1995.
11. A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of 21st Int'l Conference on Very Large Data Bases*, 1995.
12. Star Schemas and Starjoin Technology. A Red Brick Systems White Paper. 1995.
13. A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database Systems Concepts.* Mc Graw Hill, third edition, 1996.
14. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. of ACM-SIGMOD Int'l Conference on Management of Data*, 1996.

Frequent Closures as a Concise Representation for Binary Data Mining

Jean-François Boulicaut and Artur Bykowski

Laboratoire d'Ingénierie des Systèmes d'Information
Institut National des Sciences Appliquées de Lyon, Bâtiment 501
F-69621 Villeurbanne cedex, France
{Jean-Francois.Boulicaut,Artur.Bykowski}@lisi.insa-lyon.fr

Abstract. Frequent set discovery from binary data is an important problem in data mining. It concerns the discovery of a concise representation of large tables from which descriptive rules can be derived, e.g., the popular association rules. Our work concerns the study of two representations, namely frequent sets and frequent closures. N. Pasquier and colleagues designed the `close` algorithm that provides frequent sets via the discovery of *frequent closures*. When one mines highly correlated data, `apriori`-based algorithms clearly fail while `close` remains tractable. We discuss our implementation of `close` and the experimental evidence we got from two real-life binary data mining processes. Then, we introduce the concept of *almost-closure* (generation of every frequent set from frequent almost-closures remains possible but with a bounded error on frequency). To the best of our knowledge, this is a new concept and, here again, we provide some experimental evidence of its add-value.

1 Context and Motivations

One of the obvious hot topics of data mining research in the last five years has been frequent set discovery from binary data. It concerns the discovery of set of attributes from large binary tables such that these attributes are true within a same line often enough. It is then easy to derive rules that describe the data e.g., the popular association rules [2] though the interest of frequent sets goes further [8]. In this paper, we discuss the computation and the use of frequent sets considered as an interesting descriptive representation of binary table for typical rule mining processes.

When looking for a generic statement, it is possible to formulate a data mining task as a query over an intensionally defined collection of patterns [4]. Given a schema $\mathbf{R}$ for a database, let $(\mathcal{P}_{\mathbf{R}}, \mathcal{E}, \mathcal{V})$ denote the pattern domain where $\mathcal{P}_{\mathbf{R}}$ is a language of patterns, $\mathcal{E}$ is an evaluation function that defines pattern semantics, and $\mathcal{V}$ is a set of result values. Given $\mathbf{r}$, an instance of $\mathbf{R}$, $\mathcal{E}$ maps each $\theta \in \mathcal{P}_{\mathbf{R}}$ to an element of $\mathcal{V}$. Then, a mining task is the computation of the subset of $\mathcal{P}_{\mathbf{R}}$ that fulfil interestingness requirements. This can be formalized as the computation of $Th(\mathbf{r}, \mathcal{P}_{\mathbf{R}}, q) = \{\theta \in \mathcal{P}_{\mathbf{R}} \mid q(\mathbf{r}, \theta) \text{ is true}\}$ where predicate q indicates whether a sentence is considered interesting. Typically, this predicate is

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 62–73, 2000.
© Springer-Verlag Berlin Heidelberg 2000

$\mathbf{r} =$

	A	B	C	D	E
1	1	1	1	1	1
2	1	0	1	0	0
3	1	1	1	1	0
4	0	1	1	0	0
5	1	1	1	0	0
6	0	0	0	0	1

$\mathcal{E}$.support(C, **r**) = 5/6 = 0.83
$\mathcal{E}$.support(AC, **r**) = 4/6 = 0.67
$\mathcal{E}$.support($A \Rightarrow C$, **r**) = 0.67
$\mathcal{E}$.confidence($A \Rightarrow C$, **r**) = 4/4 = 1
$\mathcal{E}$.confidence($C \Rightarrow A$, **r**) = 4/5 = 0.8

Fig. 1. A binary dataset **r** and the behavior of some patterns

a conjunction of constraints that involves the evaluation function. This approach has been more or less explicitely used for various data mining tasks [13].

Example 1. Given a schema $\mathbf{R}=\{A_1, \ldots, A_n\}$ of attributes with domain {0,1} and a relation **r** over **R**, the support of a set $X \subseteq \mathbf{R}$, $\mathcal{E}$.support(X,**r**), denotes the fraction of rows of **r** that have a 1 in each column of X. Frequent set discovery in **r** consists in computing every subset from **R** such that its support is higher than a given threshold σ. Here, $\mathcal{P}_\mathbf{R}$ is $2^\mathbf{R}$, $\mathcal{V}$ is [0,1] and the predicate q is $\mathcal{E}.support(\theta, \mathbf{r}) \geq \sigma$. For instance, in Figure 1, supports of $\{C\}$ and $\{A,C\}$ in a dataset are given. Notice that we often use a string notation (e.g., AC) to denote a set of attributes. □

An explicit interestingness evaluation of all the patterns of $\mathcal{P}_\mathbf{R}$ in a dataset is not tractable in general. Though an exponential search space is concerned, frequent sets can be computed in real-life large datasets thanks to the support threshold on one hand and safe pruning criteria that drastically reduces the search space on the other hand (e.g., the so-called apriori trick [2]). However, there is still an active research on algorithms, not only for the frequent set discovery task when `apriori`-based algorithms fail (e.g., in the case of highly correlated data) but also for new related mining tasks, e.g., the discovery of maximal (long) frequent sets only [3].

Example 2. Association rules have been extensively studied since their introduction in [1]. Given the schema $\mathbf{R}=\{A_1, \ldots, A_n\}$, an association rule is an expression $X \Rightarrow Y$ where $X \subseteq \mathbf{R}$ and $Y \in \mathbf{R} \setminus X$. $\mathcal{P}_\mathbf{R}$ is the (finite) collection of such sentences. The typical "behavior" of these rules in an instance **r** over **R** is evaluated by means of two interestingness measures called "support" or "confidence". The support of a rule $X \Rightarrow Y$ is equal to the support of $X \cup Y$ (as defined in Example 1) while its confidence is equal to its support divided by the support of X. $\mathcal{V}$ is [0,1] × [0,1] and the evaluation function provides the support ($\mathcal{E}$.support) and the confidence ($\mathcal{E}$.confidence). The "classical" association rule mining task concerns the discovery of rules whose support and confidence are greater or equal to user-given thresholds, resp., σ and ϕ. The predicate q is defined as $\mathcal{E}.support(\theta, \mathbf{r}) \geq \sigma \wedge \mathcal{E}.confidence(\theta, \mathbf{r}) \geq \phi$. For example, with σ=0.5 and ϕ=0.9, $A \Rightarrow C$ is discovered in the data of Figure 1 while $C \Rightarrow A$ is not. □

In the case of association rules, left-hand and right-hand sides denote conjunctions of properties. We can consider the case of generalized rules where other boolean operators, like negation and disjunction, are allowed.

Example 3. The rule $A \wedge \neg E \Rightarrow C$ is an example of a generalized rule which might be extracted from the data in Figure 1. Its support is 0.5 and its confidence is 1. Mining such rules is very complex and we do not know any efficient strategy to explore the search space for generalized rules. □

As we are interested in very large datasets, an important issue is whether the explicit interestingness evaluation of a collection of patterns remains tractable. The answer can come from the computation of concise representations as defined in [8]. Given a database schema $\mathbf{R}$, a dataset $\mathbf{r}$ and a language of patterns $\mathcal{P}_{\mathbf{R}}$, a concise representation for $\mathbf{r}$ and $\mathcal{P}_{\mathbf{R}}$ is a structure that makes possible to answer queries of the form "How many times $p \in \mathcal{P}_{\mathbf{R}}$ occur in $\mathbf{r}$" approximately correctly and more efficiently than by looking at $\mathbf{r}$ itself. By the way, some concise representations might enable to provide exact answers.

This paper deals with two related concise representations of binary data, namely frequent sets and *frequent closures*. Not only the extraction of these representations is discussed but we also point out their specific add-value when considered as concise representations for rule mining. Beside well-studied `apriori`-based algorithms, we consider the `close` algorithm that provides frequent closures [10]. We implemented it and made experiments over real data. Furthermore, we propose the new concept of *almost-closure* and sketch the `min-ex` algorithm to mine it. The main idea here is to accept a small incertitude on set frequency since, at that cost, more useful mining tasks become tractable.

2 Frequent Sets As a Concise Representation of Binary Data

At first, we adapt the formal definition of [8] to the kind of concise representation we need. Formally, if an evaluation function $\mathcal{Q}$, a member of Θ (the class of evaluation functions), is an application from a class of structures $\mathcal{S}=\{s_i \mid i \in I\}$ into the interval [0,1], an ϵ-adequate representation for $\mathcal{S}$ with respect to Θ is a class of structures $\mathcal{H}=\{r_i \mid i \in I\}$ and an alternative evaluation function m: $\Theta \times \mathcal{H} \rightarrow [0, 1]$ such that for all $\mathcal{Q} \in \Theta$ and $s_i \in \mathcal{S}$ we have: $\mid \mathcal{Q}(s_i) - m(\mathcal{Q}, r_i) \mid \leq \epsilon$. I denotes a finite (or infinite) index set of $\mathcal{S}$.

Example 4. Let us illustrate the definition on classical concepts from programming languages. Assume $\mathcal{S}$ is a class, e.g. float, s_i is an instance of $\mathcal{S}$, e.g. 0.02, and Θ is the set of proper functions on that class, e.g. $\{sin, cos\}$. A concise representation can be the couple $(\mathcal{H}, m)$, $\mathcal{H}$ being another class, e.g. short, and m an alternative way to evaluate $\mathcal{Q}$, e.g. using a table of values of *sin* and *cos* for all angles from {0, 1, ..., 359}. Now, there is an alternative way to compute sin(x) and cos(x). Instead of s_i=0.02, we store $r_i = round(0.02 \times 360/2\pi)\ mod\ 360$, i.e., 1. When the value of $sin(0.02)$ is needed, we can use $m(sin, 1)$ that returns the value stored in the table associated to *sin*. Clearly, the result is approximate, but the error is bound and the result is known at a much lower cost. □

If the functions from Θ share a lot of intermediate results, and the number of evaluations justifies it, a concise representation can be made of the intermediate

results from which all functions from Θ can be evaluated. Such a concise representation avoids going back to the data. The alternative data representation memory requirement might be smaller as well.

Let us now consider the class $\mathcal{S}$ of binary relational schema over the set of attributes $\mathbf{R}$. Instances $s_i \in \mathcal{S}$ are relational tables. A query $\mathcal{Q} \in \Theta$ over an instance s_i of $\mathcal{S}$, denoted $\mathcal{Q}(s_i)$, is a function whose result is to be found with an alternative (ϵ-adequate) representation. $\mathcal{H}$ denotes the alternative class of structures and the counterpart of evaluations, denoted by m, must be a mapping from $\Theta \times \mathcal{H}$ into [0,1]. The error due to the new representation r_i of s_i (thus compared to the result of $\mathcal{Q}(s_i)$ on the original structure) must be at most ϵ for any instance of s_i.

Example 5. Let $\mathbf{r}$ denote a binary relation over $\mathbf{R}=\{A_1, \ldots, A_n\}$ and consider the set $\Theta=\{\mathcal{E}.\text{support}(X, \mathbf{r}) \mid X \subseteq \mathbf{R}\}$, where $\mathcal{E}.\text{support}(X, \mathbf{r})$ is the function that returns the support of X in $\mathbf{r}$ (see Example 1). Given a frequency threshold σ, let FS_σ denote the collection of all frequent sets with their supports. Let $AltSup(X, FS_\sigma)$ denote the support of a frequent set X. FS_σ and the function $m(\mathcal{E}.\text{support}(X, r), FS_\sigma) = AltSup(X, FS_\sigma)$ for $X \in FS_\sigma$, 0 elsewhere, is a σ-adequate representation for Θ over the binary relations defined on $\mathbf{R}$. □

Let us discuss the use of FS_σ as a concise representation for the rule mining task we introduced in Example 2. The support and the confidence of $X \Rightarrow Y$ are exactly known if the support of the rule is at least σ, because the first equals to $AltSup(X \cup Y, FS_\sigma)$ (since $X \cup Y \in FS_\sigma$) and the second equals to $AltSup(X \cup Y, FS_\sigma)/AltSup(X, FS_\sigma)$ (since $X \in FS_\sigma$, too). If it is not the case ($\mathcal{E}.\text{support}(X \Rightarrow Y, \mathbf{r}) < \sigma$), the support is bounded by [0, σ]. If moreover the left-hand side (X) of the rule is frequent, we can bound the confidence of the rule by [0, $\sigma/AltSup(X, FS_\sigma)$]. Otherwise, the confidence can be any number from [0, 1]). FS_σ turns to be a σ-adequate representation for rule support evaluation and a 0.5-adequate representation for rule confidence evaluation. 0.5-adequacy for confidence is clearly insufficient for most of the applications. But if we are interested only in frequent rules (support $\geq \sigma$), we get a 0-adequate representation (so an equivalent representation) for both, the support and the confidence evaluation functions. It explains the effective strategy for extracting all the potentially interesting association rules (w.r.t. frequency and confidence thresholds) from FS_σ: for each $X \in FS_\sigma$ and for each $Y \subset X$, the rule $X \setminus Y \Rightarrow Y$ is kept iff it satisfies the minimum confidence criterion.

Generalized rules (see Example 3) can be evaluated using FS_σ, too. The problem is that the collection FS_σ might not provide some of the needed supports for the computation of rule support and confidence even if the support of the rule is above the support threshold.

Example 6. Assume we want to compute the support and the confidence of the rule $A \wedge \neg E \Rightarrow D$. Applying well-known transformations, we can write the equations: $\mathcal{E}.\text{support}(A \wedge \neg E \Rightarrow D, \mathbf{r}) = \mathcal{E}.\text{support}(AD, \mathbf{r}) - \mathcal{E}.\text{support}(ADE, \mathbf{r})$ and $\mathcal{E}.\text{confidence}(A \wedge \neg E \Rightarrow D, \mathbf{r}) = \mathcal{E}.\text{support}(A \wedge \neg E \Rightarrow D, \mathbf{r}) \;/\; (\mathcal{E}.\text{support}(A, \mathbf{r}) - \mathcal{E}.\text{support}(AE, \mathbf{r}))$. These measures can be computed exactly only if A, AD, AE and ADE are frequent sets. □

If we consider several negations and disjunctions, the number of terms will increase and the need for the support of infrequent sets will increase too. Since the computation of the support of all sets is clearly untractable, infrequent conjuncts will give rise to an incertitude [8]. However, this might be acceptable for practical applications. It becomes clear that the adequacy of frequent sets as a concise representation depends on how frequent are the patterns of interest, i.e., the more a pattern is frequent, the less an incertitude will affect the result.

3 Computing Frequent Sets and Frequent Closures

The `apriori` algorithm is defined in [2] and we assume that the reader is familiar with it. It is a levelwise method based on the itemset lattice (i.e., the sets of attributes ordered by set inclusion). The algorithm searches in the lattice starting from singletons and identifies level by level larger frequent sets until the maximal frequent sets are found, i.e., the collection of sets that are frequent while none of their supersets is frequent. This collection is denoted by $Bd^+(FS_\sigma)$ and is called the positive border of FS_σ [13]. A safe pruning strategy (supersets of infrequent sets can not be frequent) has been shown to be the very efficient for the computation of FS_σ in many real-life datasets. One of the identified drawbacks of `apriori`-based algorithms is their untractability for highly correlated data mining. Data are correlated when the truth value of an attribute or a set of attributes determine the truth value of another one (in other terms, association rules with high confidence hold in it). The problem with correlated data originates from the fact that each rule with high confidence pushes the positive border back by one level for a significant part of the itemset lattice (when σ does not change). Highly correlated data contain several such rules, thus pushing back the positive border by several levels. Consequently, the extraction slows down drastically or can even turn to be untractable. An algorithm that would avoid counting support for a great part of frequent sets would accelerate the process. This is the assumption of useful algorithms like `max-miner` [3] that provides $Bd^+(FS_\sigma)$ but not FS_σ. We will consider hereafter an algorithm that avoids counting support for many frequent sets though it provides FS_σ, i.e., every frequent set and its support.

The experiment summarized in Table 1 emphasizes the influence of high correlation of data. We provide the output of the frequent set discovery tool `freddie` that implements an `apriori` algorithm. The left column corresponds to a real dataset from ANPE [1], the right one corresponds to census data (c20d10k) preprocessed at the University of Stanford [2]. We kept in both cases the first 10000 objects and for each object, their 17 first variables (each variable might be encoded in a number of binary attributes). In each column of Table 1, the first information provides the iteration counter (at level k, the level k of the itemset

[1] ANPE is the French national unemployment agency: data10K contains data about unemployed people in december 1998.

[2] ftp://ftp2.cc.ukans.edu/pub/ippbr/census/pums/pums90ks.zip.

Table 1. Mining frequent sets using `freddie` (`apriori`)

Input file : data10K			Input file : base17.txt		
Frequency threshold : 0.05			Frequency threshold : 0.05		
	Candidate sets	Frequent sets	Time (s)	Candidate sets	Frequent sets	Time (s)
Iter1 :	214	65	0.14	317	51	0.15
Iter2 :	2080	602	18.58	1275	544	14.60
Iter3 :	2991	2347	78.76	3075	2702	92.12
Iter4 :	5738	4935	223.95	8101	7940	376.87
Iter5 :	7203	6623	367.86	15454	15365	965.41
Iter6 :	6359	5957	391.79	20720	20705	1564.63
Iter7 :	3733	3558	257.88	19973	19968	1777.45
Iter8 :	1395	1359	105.20	13859	13857	1429.21
Iter9 :	304	302	23.13	6811	6811	798.39
Iter10 :	32	32	2.70	2277	2277	292.68
Iter11 :	1	1	0.48	479	479	58.83
Iter12 :	No more.			54	54	5.89
Total :	34836	25781	1470.47	2	2	0.74
				Iter14 : No more.		
				Total : 97080	90755	7376.97

lattice is processed). Then, we get the number of candidates, the number of frequent sets and finally the duration of the iteration (CPU time).

The "independance analysis" of the data has shown that ANPE data are slightly correlated while census data are highly correlated. However, the average level of correlation in ANPE data is not low. Typical basket analysis data are much less correlated and would bring down the execution time to a few minutes (and the number of frequent sets would certainly be smaller for $\sigma = 0.05$).

The problem is clearly that a user might want to mine (highly) correlated data with rather low support thresholds while `apriori`-based algorithms become untractable (time, memory) in that cases.

`close` is an algorithm that computes frequent closures in binary data [10]. A set X is a closure in $\mathbf{r}$ when there is no attribute in $\mathbf{R} \setminus X$ that is always true when attributes in X are true. In other words, for each property p not in X, there is a tuple in $\mathbf{r}$ that has all properties of X and does not have the property p. A closure is called a frequent closure when its support in the database is greater than a given threshold σ.

Example 7. In the data from Figure 1, BC is closed while BD is not closed. Indeed, the objects 1 and 3 (the only ones that verify B and D) verify A and C, as well. Furthermore, if σ=0.6, BC is a frequent closure in that data. □

By reducing the number of candidates considered during the extraction (the lattice of closures is generally quite much smaller than the lattice of itemsets, see for instance Figure 2 on the left), `close` can be more efficient than `apriori`. It is straightforward to derive all the frequent sets and their supports from frequent closures.

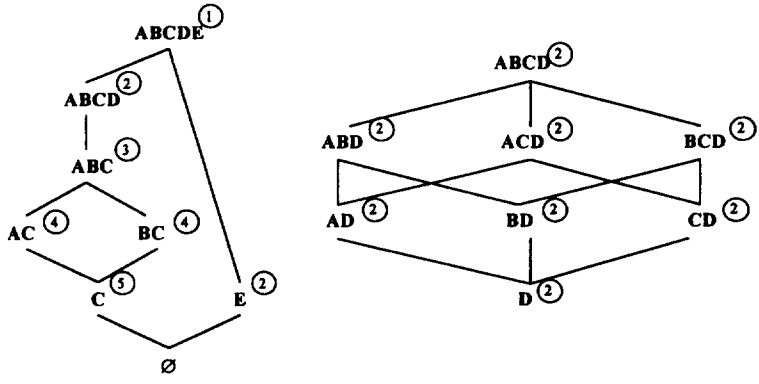


Fig. 2. Closed set lattice (left) and sub-lattice of itemset lattice w.r.t. generator D (right) for the data from Example 1

We now sketch the `close` algorithm and introduce our implementation `close`$_2$. Formal definitions and proofs of properties about `close` are in [10]. Mining closures as a formal basis for association rule mining has also been suggested in [12] though no algorithm was proposed in that paper.

Let FC_σ denote the collection of all frequent closures and their supports. The positive border of FC_σ, $Bd^+(FC_\sigma)$, is the set containing all frequent closures for which no superset of each of them is in FC_σ. It has been proven that, for a given dataset, $Bd^+(FC_\sigma) = Bd^+(FS_\sigma)$.

There are two properties of the itemset lattice on which substantial optimisations can rely. First, the supports of a set and of its closure are the same (see the right part of Figure 2 for an example derived from Example 1). Thus, once identified the closure of a set to be different from this set, we can exclude the closure and all intermediate sets from the support counting procedure since they all have the same support. The sets that go through the support counting procedure are called *generators*. In Figure 2 on the right, it is emphasized that counting the support of generator D, whose closure is $ABCD$, enables to derive the support for the whole sub-lattice. Second, if the closure of X is $X \cup C$, the closure of $X \cup Y$ is a superset of $X \cup Y \cup C$. These properties are used as a base of a safe pruning strategy integrated in `close` [10].

In our implementation `close`$_2$, the extraction of frequent sets is performed in two steps. The first step extracts frequent closures from a binary relation. The extracted closures correspond to all generators. There may be some duplicates, in terms of closures, because different generators may have a same closure. The second step takes that collection of frequent closures, removes duplicates, stores FC_σ set and derives FS_σ. In Table 2, we compare the execution of `close`$_2$ with `freddie` on ANPE and census data. The given time is the average CPU time for 2 executions. For `close`$_2$, the time of each step is given. The I/O overhead is provided as the number of scans on the data. We notice that the relative advantage of `close`$_2$ over `freddie` is much higher in case of highly correlated data. However, in both cases, the use of `close`$_2$ is worthwhile.

Table 2. Comparison of `freddie (apriori)` and `close`$_2$

	`freddie (apriori)`			`close`$_2$		
Dataset/σ	Time (s)	FS_σ	DB scans	Time (s)	FC_σ	DB scans
ANPE/0.05	1463.9	25 781	11	69.2/6.2	11 125	9
census/0.05	7377.6	90 755	13	61.7/25.8	10 513	9
ANPE/0.1	254.5	6 370	10	25.5/1.1	2 798	8
census/0.1	2316.9	26 307	12	34.6/6.0	4 041	9
ANPE/0.2	108.4	1 516	9	11.8/0.2	638	7
census/0.2	565.5	5 771	11	18.0/1.1	1 064	9

As it is possible to generate FS_σ from the corresponding FC_σ and $||FS_\sigma|| \geq ||FC_\sigma||$, FC_σ can be considered as a concise representation of the binary relation which is more compact than FS_σ, without any loss of information. Beside efficiency, notice that the postprocessing of frequent closures to get rules can also give rise to a faster computation of useful rules. A first study in that direction concerns the computation of non redundant rules [11].

4 A New Concise Representation: Mining Almost-Closures

This section concerns the concept of almost-closure in binary data. To the best of to our knowledge, this is an original concept. Details about the formalization and the algorithm are available in [6,5].

A fundamental property of set lattices which is used in `close`, is that the same support of the sub-lattice's bottom and top implies the same support for all sets of that sub-lattice. The more the data is correlated (many association rules with confidence 1), the more the collection of frequent closures is compact compared to the collection of frequent sets. We decided to relax the constraint equality of supports, which seems to be a very exigent one, with an "almost-equality" constraint. The new algorithm, called `min-ex`, does not require any association rule with confidence 1 to be present in the mined data. Instead, it can take advantage of a correlation even if it is approximate (the confidence of association rules holding in the data should be however close to 1). These situations might correspond to exceptions in regular behaviours and/or to erroneous tuples that survived preprocessing steps. We expect that, in case of real-life data mining, we will remove much more candidates (w.r.t. `close`) from the support counting procedure, given that `min-ex` pruning strategy is similar to `close` pruning strategy. The trade-off consists in accepting a small incertitude on supports though being able to mine correlated data with lower frequency thresholds. In the following, we consider that the support of a set is the (absolute) number of objects (tuples) in which all the attributes of the set are true. This is different from the definition in Example 1.

Formally, if X (an itemset) "occurs" in t objects within the database, we say that an attribute A is in the almost-closure of X if the support of $X \cup \{A\}$ is at least $t - \delta$ (δ should be small, not to loose the practical relevancy of the

extracted information). The almost-closure of X is the set containing all such attributes. Conceptually, a closure is a special case of an almost-closure when δ=0.

Example 8. In data from Figure 1, considering the generator C, one finds that A and B are in the almost-closure of C for δ=1 while none of them was in its closure. □

Now, let us explain where the incertitude comes from. Assume that the almost-closure of X equals to $X \cup \{A, B, C\}$. Let the support of X be s_X, and the supports of $X \cup \{A\}$, $X \cup \{B\}$ and $X \cup \{C\}$ be respectively $s_X - s_A$, $s_X - s_B$ and $s_X - s_C$ where s_A, s_B and s_C are positive numbers lower than δ. We have considered two possibilities for output content. The first stores for each frequent almost-closure: generator items (elements of X, in the example), generator support (s_X) and almost-closure's supplement items (A, B and C). The second adds to each item a from the almost-closure supplement the difference of support between X and $X \cup \{A\}$ (this difference is called miss-number hereafter). In our example that part corresponds to s_A, s_B and s_C. These values have to be known, because to decide if an item is in the almost-closure, they must be at hand. Miss-numbers are values of miss-counters at the end of the corresponding database pass.

The fact, that, for instance, B and C are in the almost-closure of X only implies that they occur almost always with X. Assume that we are in the second case of output (miss-numbers stored). From the supports of X, $X \cup \{B\}$ and $X \cup \{C\}$ we can not infer the support of $X \cup \{B, C\}$, because we do not know if the misses occurred on the same objects (support would be $s_X - max(s_A, s_B)$) or on disjoint ones (support would be $s_X - s_A - s_B$). All intermediate cases are allowed, too. Storing miss-numbers greatly improves the precision of the resulting supports, above all when they are small, compared to δ. Therefore, we choose this solution, even if it increases the volume of output (in terms of quantity of information, not in terms of number of elements). FaC_σ denotes the collection of all frequent almost-closures for threshold σ and is the output of `min-ex`.

An important property about closures has been preserved. Still, if the almost-closure of X is $X \cup C$, the almost-closure of $X \cup Y$ is a superset of $X \cup Y \cup C$. Let us prove it. Attribute A is in the almost-closure of X iff $\mathcal{E}.\text{support}(X, \mathbf{r}) - \mathcal{E}.\text{support}(X \cup \{A\}, \mathbf{r}) \leq \delta$. In other words, the number of objects that have all properties of X and do not have the property A is at most δ. Clearly, the number of objects satisfying a set of properties can not grow if we enforce that property with a new constraint. Therefore, the number of objects that have all properties of X and all properties of Y and do not have the property A can not be greater than δ. So, all elements of the almost-closure of X (i.e. C) must be present in the almost-closure of $X \cup Y$.

This property may be used as a basis of an efficient safe pruning strategy, analogously to the pruning strategy of `close`. We have been looking for such a strategy. The one implemented in the actual implementation of `min-ex` seems to be reliable [6]. However, in spite of numerous tries, we did not establish a proof that it is safe. We have not found either a counterexample. We checked the

completeness in our practical experiments. However, proving the incompleteness or the completeness of our algorithm remains an open problem though it does not prevent its use for practical applications.

Deriving frequent sets from frequent almost-closures is as straightforward as for `close`. The difference is that now there is an incertitude on the support of some frequent sets.

The sub-lattices (corresponding to almost-closures) of which the support range, due to δ, crosses the threshold is kept in the result set, leading to the collection FaC_σ that enables to derive a superset of FS_σ. This is a safety measure: we do not want to prune out sub-lattices of which some itemsets are known to be frequent, for the sake of completeness.

We did several experiments using `min-ex` on census and ANPE datasets (see Table 3). A first remark is that it confirms that `close` and `min-ex` with δ=0 are functionally equivalent. In the case of $\texttt{close}_2$, the reduction of the size of FC_σ w.r.t. the corresponding FS_σ highlights the tight-correlation level (relative number of rules with confidence 1) of the data. In the same way, the further reduction of output (——FaC_σ—— compared to ——FC_σ——) for different values of δ, points out the loose-correlation level (relative number of association rules that are nearly "logical" ones).

Let us now discuss the add-value of `min-ex` w.r.t. `close` for highly correlated data mining like census data mining. First of all, we must recall that a too high value of δ might provide a "fuzzy" FaC_σ collection, leading to, e.g., rules with too high incertitude on evaluation functions.

Consider the CPU time needed by the extraction of FaC_σ. It has been more than halved (census data) for δ=6 and the tested frequency thresholds. Next, the I/O activity (number of database passes) has been reduced, an important criterion if the I/O turns to be a bottleneck of the system. A third advantage is that the output collection size has shrunk and we assume that further subsequent knowledge extraction steps will be faster.

Another way to demonstrate the add-value of `min-ex` can be derived from Table 3. We can extract the following concise representations of census data: either $FC_{0.01}$ with `close` or $FaC_{0.005}$ with `min-ex` and $\delta = 2$. It took the same time (154.3 vs. 155.2 sec., 10 passes for both executions) and we got a similar-sized output collection (52166 vs. 55401 itemsets). It is possible without incertitude ($FC_{0.01}$) or with a very good precision (δ=2) on the frequent set supports ($FaC_{0.005}$). The difference is that, using `min-ex`, we gained knowledge about all phenomena of frequency between 0.5% and 1% at almost no price. However, we must notice that in case of uncorrelated data, the memory consumption and CPU load due to maintaining miss-counters may affect the performances (See in Table 3 the extraction time evolution for ANPE/σ=0.05). Only, with a significant reduction of number of candidates (thus only in case of correlated data), the memory consumption will recover (e.g., see ANPE/σ=0.005 or census/σ=0.05).

Applications. A promising application of `min-ex` would be to enable the discovery of repetitive but scarce behaviours. Another application concerns generalized rule mining. Generalized rules, if generated from FS_σ, have an incertitude on

Table 3. Evaluations of implementations $\texttt{close}_2$ and `min-ex`

	$\texttt{close}_2$			`min-ex`			
Dataset/σ	Time (s)	FC_σ	DB scans	δ	Time (s)	FaC_σ	DB scans
ANPE/0.005	816.7	412 092	11	0	851.3	412 092	11
				2	759.5	265 964	11
				4	639.7	182 829	10
				6	553.0	135 136	10
census/0.005	197.8	85 950	10	0	216.2	85 950	10
				2	155.2	55 401	10
				4	118.4	39 036	8
				6	98.5	29 848	8
ANPE/0.01	421.8	161 855	11	0	450.4	161 855	11
				2	466.8	130 765	11
				4	445.1	104 162	10
				6	416.4	84 318	10
census/0.01	154.3	52 166	10	0	166.2	52 166	10
				2	124.9	33 992	10
				4	95.0	24 109	8
				6	79.0	18 822	8
ANPE/0.05	69.2	11 125	9	0	71.5	11 125	9
				2	79.7	11 066	9
				4	85.3	10 931	9
				6	88.4	10 588	9
census/0.05	61.7	10 513	9	0	64.4	10 513	9
				2	50.2	7 294	9
				4	38.2	5 090	8
				6	32.2	4 086	8

measures like support and confidence due to unknown infrequent set supports [8]. Using `min-ex`, it is possible to reduce the bounds of error on evaluation value by supplying the support value for many more itemsets. The incertitude introduced by `min-ex` to some terms of generalized rule evaluation functions can be negligible (w.r.t. function result) compared to the contribution made by the larger number of known terms. Another interesting use is when an approximate result of the data mining step is sufficient. For instance, consider the "sampling" algorithm [7] during its "guess" phase. This phase is supposed to provide an approximation of the collection of frequent sets. An error is inherent to the use of sampling. If we keep the error introduced by the use of almost-closures negligible against the error due to sampling, the guess will be as good as before, but will be computed faster.

5 Conclusion

We studied several concise representations of binary data when data mining processes make use of set support (e.g., when looking for association rules). We studied the `close` algorithm and beside its introduction in [10], we provide a new

implementation and experimental evidences about its add-value for the concise representation of (highly) correlated data. It has lead us to the definition of the concept of almost-closure and, here again, we provided experimental evidences of its interest when we are looking for concise representation in difficult cases (correlated data and low frequency thresholds). The discovery of almost-closed frequent sets gave rise to tricky problems w.r.t. the completeness of the mining task. Completeness of `min-ex` remains an open problem at that time and we are currently working on it.

Acknowledgements. The authors thank H. Toivonen from the University of Helsinki for letting us use the `freddie` software tool and the Rhône departmental direction of ANPE who provided data. Last but not least, we want to thank C. Rigotti for stimulating discussions.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In: Proc. *SIGMOD'93*, Washington DC (USA), pages 207 – 216, May 1993, ACM Press.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In: *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328, 1996, AAAI Press.
3. R.J. Bayardo. Efficiently mining of long patterns from databases. In: Proc. *SIGMOD'98*, Seattle (USA), pages 85 – 93, June 1998, ACM Press.
4. J-F. Boulicaut, M. Klemettinen, and H. Mannila. Modeling KDD processes within the Inductive Database Framework. In: Proc. *DaWak'99*, Florence (I), pages 293 – 302, September 1999, Springer-Verlag, LNCS 1676.
5. J-F. Boulicaut, A. Bykowski, and C. Rigotti. Mining almost-closures in highly correlated data. Research Report LISI INSA Lyon, 2000, 20 pages.
6. A. Bykowski. Frequent set discovery in highly correlated data. Master of Science thesis, INSA Lyon, July 1999, 30 pages.
7. H. Toivonen. Sampling large databases for association rules. In: Proc. *VLDB'96*, Mumbay (India), pages 134 – 145, September 1996, Morgan Kaufmann.
8. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In: Proc. *KDD'96*, Portland (USA), pages 189 – 194, August 1996, AAAI Press.
9. H. Mannila. Inductive databases and condensed representations for data mining. In: Proc. *ILPS'97*, Port Jefferson, Long Island N.Y. (USA), pages 21 – 30, October 1997, MIT Press.
10. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, Volume 24 (1), pages 25 – 46, 1999.
11. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set discovery of small covers for association rules. In: Proc. *BDA'99*, Bordeaux (F), pages 53 – 68, October 1999.
12. M. Zaki and M. Ogihara. Theoretical foundations of association rules. In: Proc. *Workshop post-SIGMOD DMKD'98*, Seattle (USA), pages 85 – 93, June 1998.
13. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241 – 258, 1997.

An Optimization Problem in Data Cube System Design

Edward Hung, David W. Cheung, Ben Kao, and Yilong Liang

Department of Computer Science and Information Systems,
The University of Hong Kong, Hong Kong.
{ehung,dcheung,kao,ylliang}@csis.hku.hk

Abstract. In an OLAP system, we can use data cubes (precomputed multidimensional views of data) to support real-time queries. To reduce the maintenance cost, which is related to the number of cubes materialized, some cubes can be merged, but the resulting larger cubes will increase the response time of answering some queries. In order to satisfy the maintenance bound and response time bound given by the user, we may have to sacrifice some of the queries and not to take them into our consideration. The optimization problem in the data cube system design is to optimize an initial set of cubes such that the system can answer a maximum number of queries and satisfy the bounds. This is an NP-complete problem. Approximate algorithms Greedy Removing and 2-Greedy Merging are proposed. Experiments have been done on a census database and the results show that our approach is both effective and efficient.

1 Introduction

1.1 DSS and OLAP

With the advancement of data warehousing technology, corporations are building their decision support systems (DSS) on large data warehouses. In order to support on-line analytical processing (OLAP), the system is required to answer queries with a fast response time. However, queries are usually about summarization information, so the system needs to scan almost the entire database, giving a very poor response time. One efficient approach to reduce the response time is to translate frequently asked queries to *data cubes* or simply *cubes*, which are precomputed multi-dimensional views of the data in the data warehouse.[3] Once the cubes are built, answers to the queries can be retrieved from the cubes in real time.

An OLAP system can be modeled by a *three-level architecture* that consists of: (1) a query client; (2) a data cube engine; and (3) a data warehouse server (Figure 1). The bottom level is a data warehouse built on top of one or more source operational DBMSs. It needs to support fast aggregations by using different indexing techniques such as bit-map indices and join indices [6,7]. The middle level contains a set of cubes generated from the data warehouse. The top

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 74–85, 2000.
© Springer-Verlag Berlin Heidelberg 2000

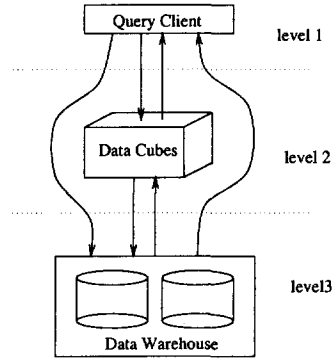


Fig. 1. Three-level Architecture of a Data Cube System

level is a query client, which supports DSS queries and allows user to browse the data cached from the cubes (like slicing and dicing). A query submitted to the client interface level, after being checked against the cube set, will be directed to the middle level if it can be answered by the cubes there; otherwise, the query is passed to the bottom level, from which the results can be derived. Since cubes store pre-computed results, it is much faster to answer queries with cubes than with the data warehouse.

Various studies have been made on the three levels of OLAP system. For the cube level, the main research focuses on the two issues: (1) how to compute aggregates from a base cube efficiently, and (2) how to store a cube. However, our previous research study [2] has shown that the key to the design of a query-efficient OLAP system lies on the design of a good cube set. We remark that the initial cube set, with one cube being tailor-made to answer each query, gives the best query performance. The OLAP system would be able to support real-time responses if the cube level can answer all the queries. Unfortunately, more the cubes materialized, higher is the maintenance cost because it usually takes more number of scanning on the database to compute aggregates during maintenance. Thus, materializing all possible cubes is clearly impractical. We call the maximum number of cubes materialized as the *maintenance bound*. We can reduce the cube number by selecting several disjoint subsets of cubes and merging them into some larger cubes. As a result, a smaller cube set can be obtained. This improves the maintenance cost at the expense of increasing query response time because some queries processed using merged cubes generally take longer time than using the original smaller cubes as the resulting cube is in general larger than any one of the component cubes. Therefore, there is a trade-off between query performance and cube maintenance cost.

Since the OLAP system needs to support real-time responses, a *response time bound* is necessary, i.e., the sum of response time to answer a selected set of frequent queries using the resultant cube set should not exceed the bound given by the user. However, sometimes the maintenance bound and the response time

bound that the user gives are so strict that there does not exist any solution even after considering all possible merging of cubes. In that case, we have to sacrifice some of the queries, remove them from our selected set of queries and not to consider the response time of answering them. Also, the corresponding cubes are removed from our initial set. As a result, we may be able to find a solution from the smaller set of cubes, which satisfies the bounds. Our problem now is how to choose a minimum number of cubes to remove, and how to merge the remaining cubes so that both of the maintenance bound and the response time bound are satisfied. This is an NP-complete problem, and what we can do is to develop some efficient and effective approximate algorithms so that a solution with acceptable performance can be found in an acceptable execution time.

1.2 Data Cube System Design

Given the user query requirements, namely, a set of frequently asked queries, a maintenance bound and a response time bound, our goal is to derive a data cube set that satisfies both of the bounds. Our approach to this data cube system design problem is a two-phase process:
(1) Design of the Initial Data Cube Set

The first phase is to derive an initial set of data cubes from the set of frequently asked queries. This set is called the *initial data cube set.* The answer of each query can be retrieved directly and efficiently from a cube in this initial set.
(2) Optimization of Data Cube System Design

The second phase is an optimization of the initial data cube set so that the maintenance bound and the response time bound are satisfied by removing a minimum number of queries and merging the remaining cubes.

1.3 Related Works

Several papers have been published on data cube implementation. Cube selection algorithms have been proposed in [5,8]. Our optimization approach differs from these previous works in two aspects. First, cube selection assumes that there is one root base cube encompassing all the attributes in all the queries and that some queries are associated with the root base cube. As a result, a cube selection would always include the root base cube in the answer. In a general DSS such as TPC-D [9], we do not anticipate many frequent queries that involve all the attributes; hence, cube selection is not suitable for solving our problem. Secondly, most cube selection algorithms start from the base cube at the top level and determine what cubes deducible from it should be implemented so that queries on the aggregates in the base cube can be answered efficiently. Tackling a very different problem, cube optimization tries to merge the cubes in an initial set in a bottom-up manner to generate a set of cubes which has an optimal performance for a given set of frequently-asked queries. The search space of the optimization problem is in general much larger because of the large numbers of

attributes in the initial data cube set. In short, cube selection algorithms are for cube implementation but not for cube optimization.

1.4 Organization of Paper

After introducing the optimization problem in data cube system design and our approach to solve it, we will discuss the optimization problem in details in Section 2. The optimization phase in our approach can be divided into two levels, which will be discussed in Section 3 and 4. The performance study is described in Section 5. Finally we give a conclusion in Section 6.

2 Data Cube System Design Optimization

2.1 Search Space of an Optimal Set

In this paper, we assume that the requirements of an OLAP system is captured in a set of frequent queries. We use Q to denote the initial cube set derived from the queries. For example, if a query involves the attributes a, b, c, a three-dimensional cube on these attributes is included in Q. Before defining the optimization problem, let us first discuss the search space of the problem.

To simplify the problem, we assume that the database in the data warehouse is represented by a star schema [1]. Attributes in the queries come from the fields of the dimension and fact tables, which may contain many attributes. As a result, the number of attributes (dimensions) needed to be considered in a data cube system design is much more than the number of dimension tables. In TPC-D [9], 33 attributes need to be considered.

In [5], the notion of a *composite lattice* is used to integrate multi-hierarchical dimensions with the lattice of aggregates in a data cube. Assume that $A = \{a_1, a_2, \cdots, a_n\}$ is the set of all the attributes on which query can be posted. Any subset of A can be used as the dimension attributes to construct a cube. The composite lattice $L = (\mathcal{P}(A), \prec)$ is the lattice of cubes constructed from all the subsets of A. ($\mathcal{P}(A)$ is the power set of A.) The cube associated with the set A is the root of the lattice L. For two different cubes c_1, $c_2 \in L$, the *derived from* relationship, $c_1 \preceq c_2$, holds if c_1 can be derived from c_2 directly or by aggregation. For example the cube $c_1 = [part,\ year]$ can be derived from $c_2 = [part,\ customer,\ date]$. The lattice L is the search space of our optimization problem. As we have mentioned, n is large in general. Thus, the search space L of the optimization problem is enormous.

2.2 Problem Definition

Given an initial data cube set Q, a search space L, a maintenance bound MB, and a response time bound RTB, the optimization problem in data cube system design is defined in Table 1.

Table 1. Optimization Problem in Data Cube System Design

Objective: Find $P \subset Q, C \subset L$ such that $\|P\|$ is maximum and $Cost(P,C) \leq RTB$
Constraint: $\forall p \in P, \exists c \in C$, such that $p \preceq c$ and $MC(C) \leq MB$

Each cube in P is derived from a particular query, so P contains a set of cubes that can be used to directly answer some certain queries. $Cost(P,C)$ is the total query cost of answering the queries associated with P by using the cubes in C. The constraint states that any frequent query p can be answered by some cube c in C. $MC(C)$ is the total maintenance cost of C. Therefore, the problem is to choose a *maximal* subset P (and the corresponding queries) from the initial set Q, and a set of cubes C in L, such that the cost of answering the queries (associated to P) by using C is under the response time bound RTB and the cost of maintaining C does not exceed the maintenance bound MB.

For simplicity, we assume that the weights and the number of queries associated with each $q \in Q$ are the same. Then we can use $q \in Q$ to represent both a cube in the initial set and the queries associated with it. Since we do not want to make any assumption on the implementation of the cubes and the structure of the queries, a good measure of $Cost(P,C)$ is the linear cost model suggested in [5]. In that model, for cubes p, c, if $p \preceq c$, then *the cost of deducing the answers for the query p from the cube c is linear to the number of data points in c.* We use $S(c)$ to denote the number of data points in c. With respect to the cost model, many sampling and analytical techniques can be used to estimate the number of data points. For each query $p \in P$, we need to determine a minimum-cost (size) cube $c \in C$ from which the answer of p can be deduced. We use $F_C(p)$ to denote the smallest cube in C for answering p, i.e., $F_C(p)$ is a cube in C such that $p \preceq F_C(p)$ and $\forall x \in C$, if $p \preceq x$, then $S(F_C(p)) \leq S(x)$. * We can now define $Cost(P,C)$ by the formula:

$$Cost(P,C) = \sum_{p \in P} (S(F_C(p))) \tag{1}$$

Without assuming any implementation method, we use the following measure to determine the maintenance cost $MC(C)$ in the problem definition.

$$MC(C) = |C|, \text{ i.e., the number of cubes in } C. \tag{2}$$

The bound MB is the maximum number of cubes in the data cube system. **

* We use the condition $S(F_C(p)) \leq S(x)$ instead of $F_C(p) \preceq x$, because $F_C(p)$ needs to be the one that has the minimum size.

** A more general approach is to define $MC(C)$ as the total size of the cubes in C. In this paper, we have taken the more restricted model in order to develop a solution first for the simpler case. Results for the general case are being written in [4].

Our approach to the optimization problem consists of two levels: the *query level* (the higher level) and the *attribute level* (the lower level). In this paper, two algorithms *Optimal Removing* and *Greedy Removing* (of queries or cubes) are described for the query-level optimization. Two algorithms *Optimal Merging* and *2-Greedy Merging* are described for the attribute-level optimization. Our suggestion is to use Greedy-Removing at the query level and 2-Greedy Merging at the attribute level. The details will be discussed in the following sections.

3 Query-Level Optimization

In this section, we discuss the query-level optimization. Given an initial data cube set Q, optimization at the query level is to find a subset P of Q such that the set C obtained by merging some cubes in P satisfies the maintenance bound (MB) and response time bound (RTB), and the number of data cubes in P (or the size of P) is maximum, i.e., the number of queries removed is minimum.

3.1 Optimal Removing

```
    /* input: L, Q, MB, RTB; output: P, C */
1)  j = |Q|;
2)  while(j ≥ 1) {
3)      S = AllSubsets(Q, j);
4)      for(P ∈ S) do {
5)          C = AO(L, P, MB)
6)          if (Cost(P, C) ≤ RTB) return P, C;
7)      }
8)      j = j − 1;
9)  }
10) return "The response time bound is too strict."
```

Fig. 2. The algorithm Optimal Removing

Optimal Removing (OR) finds the optimal removal of cubes. The outline of OR is shown in Figure 2. The algorithm starts with the initial set Q. The loop (lines 2-9) tries all possible combinations of j queries starting from keeping all cubes in the initial set Q. In line 3, $S = AllSubsets(Q, j)$ assigns S with all subsets of Q with the number of queries equal to j. In the loop in lines 4-7, for each set P in S, the attribute-level optimization ($AO(L, P, MB)$) (which will be discussed in the next section) refines P to C (which satisfies the maintenance bound) (line 5), and then it is checked whether the refined set C satisfies the response time bound or not (line 6). If yes, OR returns P. If none of them is a solution, we decrease the cube number by one. The iteration of loop in lines 2-9 stops when a solution is found or after removing all cubes.

If the number of the initial cubes is n, then the number of ways of keeping n cubes is ${}_nC_n$. Then in the next iteration, the number of ways of keeping $(n-1)$ cubes is ${}_nC_{n-1}$, etc. With respect to the number of attribute-level optimization done, in the worst case, the time complexity of Optimal Removing is ${}_nC_n + {}_nC_{n-1} + \ldots + {}_nC_m = O(n^{\lceil \frac{n}{2} \rceil})$ if $m \leq \frac{n}{2}$ or $O(n^{n-m})$ if $m > \frac{n}{2}$. The optimal method is too time-consuming in practice.

3.2 Greedy Removing

```
    /* input: L, Q, MB, RTB; output: P, C */
1)  j = |Q|;
2)  P = Q;
3)  C = AO(L, P, MB)
4)  if (Cost(P, C) ≤ RTB) return P, C;
5)  while(j ≥ 1) {
6)     for all (p ∈ P) do {
7)        C = AO(L, P − p, MB)
8)        if (Cost(P − p, C) ≤ RTB) P = P − p, return P, C;
9)     }
10)    P = P − p which gives the minimum Cost(P-p, C) ;
11)    j = j − 1;
12) }
13) return "The response time bound is too strict."
```

Fig. 3. The algorithm Greedy Removing

Greedy Removing (GR) is a simple, efficient, but still very effective method. The outline of GR is shown in Figure 3. P is the set of cubes kept. First we check whether keeping all cubes ($P = Q$) gives a solution. If not, we check whether there is a solution among all possible ways of removing one cube from P. We choose to remove a cube p from P (i.e., $P = P-p$) if the resulting cube set $(P-p)$ gives us a minimum response time among all others. The above is repeated until either the response time bound is satisfied or all cubes are removed.

If the number of initial cubes is n, the number of ways of removing one cube from them is n. After we have removed one cube, the number of ways of removing another data cube is $n-1$, etc. With respect to the number of attribute-level optimization done, in the worst case, the time complexity of Greedy Removing is $1 + n + (n-1) + \ldots + (m+1) = O(n^2)$. Greedy Removing is very efficient compared with Optimal Removing. It is also very effective, as we will see in the performance study section.

4 Attribute-Level Optimization

In this section we discuss how we optimize a given set of data cubes at the attribute level. Given a set of data cube set P, optimization at the attribute

level is to refine the set P to C so that the sum of the response time answering all queries associated to P by C is minimum with the maintenance cost within the maintenance bound MB. Essentially, this means that we need to divide the cubes in P into a number of groups. Cubes in each group are then merged into one single cube.

4.1 Optimal Merging

A straight-forward brute-force approach would try all possible groupings such that the number of groups are within the maintenance bound. We call this approach *Optimal Merging (OM)*. However, it is only necessary to consider those cases in which the number of groups is equal to the maintenance bound. ***

We outline OM in Figure 4. In line 1, $TOP(L)$ returns the top node of the

```
    /* input: L, P, MB; output: C */
1)  C = TOP(L);
2)  G = AllGroupings(P, MB);
3)  for all g ∈ G, do
4)      if (Cost(P, g) < Cost(P, C)) C = g;
5)  return C;
```

Fig. 4. The algorithm Optimal Merging

lattice L, which contains all attributes, and so this cube has the largest size. C is initialized as the top node. In line 2, the function $AllGroupings(P, MB)$ considers all possible groupings of cubes in P with the number of groups equal to MB. For each grouping, cubes in every group are merged into one cube. The resultant sets of cubes of all groupings are returned to G (G is a set of cube sets). Therefore, each element $g \in G$ contains a cube set. In line 3, we run the loop (lines 3 to 4) to consider the response time of every cube set g in G ($Cost(P, g)$) in order to find a cube set whose response time of answering query set P is minimum. The resultant cube set C is returned in line 6.

If there are initially n cubes, and the maintenance bound is m, then in the worst case, the total number of ways of grouping the n cubes into m groups is $O(m^{n-m})$. The time complexity is too large in practice.

4.2 2-Greedy Merging

In 2-Greedy Merging (2GM), starting from a given cube set P, we try to merge cubes step by step in a bottom-up Greedy approach. The outline of 2-Greedy Merging is shown in Figure 5. In each loop (lines 2 to 5) of the algorithm, we

*** It is not necessary to consider other cases because the number of groups should not exceed the maintenance bound, and if the number of groups is less than the maintenance bound, it is impossible for its total response time to be smaller than that of the optimal solution.

```
    /* input: L, P, MB; output: C */
1)  C = P;
2)  while MC(C) > MB do {
3)      SelectCubes(D ⊆ C, A ⊆ L − C) such that α(C, D, A) is maximal,
            |D| = 2 and |A| = 1;
4)      C = C ∪ A − D;
5)  }
6)  return C;
```

Fig. 5. The algorithm 2-Greedy Merging

select two cube sets: D with two cubes and A with one cube. The cubes in D are removed from C, and the cube in A is added into C such that each cube p in the input set P can be derived from a cube in the new C. Therefore, the maintenance cost $MC(C)$ (cube number) would decrease and the query cost would increase. The algorithm terminates when the maintenance bound is satisfied. In each iteration, we want to choose a new C such that the increment in the query cost is small. Thus, we define our evaluation function α by the following formula.

$$\alpha(C, D, A) = \frac{1}{Cost(P, C^+) - Cost(P, C)} = \frac{1}{\sum_{p \in P}[S(F_{C^+}(p)) - S(F_C(p))]} \quad (3)$$

where $C^+ = C \cup A - D$ is the new C. The denominator is the increment in the query cost. $F_{C^+}(p)$ is the smallest cube in C^+, which can answer p.

Let 2GM begin with n cubes. It first considers ${}_nC_2$ ways of choosing a cube pair for merging. There are $n-1$ cubes left in the second iteration, and the algorithm considers ${}_{n-1}C_2$ ways of merging two cubes. It stops after only m cubes are left, where m is the maintenance bound. Thus in the worst case, the time complexity of the algorithm is ${}_nC_2 + {}_{n-1}C_2 + \ldots + {}_{m+1}C_2 = O(n^3)$, which is obviously much lower than that of Optimal Merging ($O(m^{n-m})$). Although it does not guarantee an optimal solution, its performance is near-optimal as we will see in the performance study section.

5 Performance Study

5.1 Experimental Setup

Experiments were done to test the effectiveness and efficiency of the optimization algorithms proposed at the attribute and query levels. The database used is a Hong Kong census data with 62010 tuples. Each tuple has 15 integer fields, with ranges from tens to tens of thousands. The queries were generated randomly with the probability of $\frac{1}{3}$ for a particular attribute to appear in a query. The experiments were done on a Sun Enterprise Ultra 450 with 4 UltraSPARC-II CPU running at 250MHz.

5.2 Attribute Level

The range of query number is from 5 to 15, and that of attribute number involved is from 10 to 15. The maintenance bound is generated randomly around three-tenths to eight-tenths of the query number.

Let the response time of answering the given queries using the cubes refined from 2-Greedy Merging (2GM) be GRT, and that from Optimal Merging (OM) be ORT, then the *performance ratio (PR)* is defined as $\frac{GRT}{ORT}$. From Table 2 (a), 54 out of 66 cases (over 81 percents) give a PR of 1. That is to say, the performance of answering the set of frequent queries using the solution of 2GM is the same as that of the OM. PR of other results are very close to 1. The average PR is 1.003661, which is very close to 1. The maximum (worst) PR so far discovered is 1.1274. Table 2 (b) shows that the execution time of 2GM is less than 1 second in all cases and grows at a sub-cubic rate with the query number. Our experiments also show that the execution time of OM grows exponentially with the query number. 2GM is thus a very efficient and effective algorithm for attribute-level optimization.

Table 2. Table of (a) Performance Ratio of 2GM vs OM, (b) Execution time of 2GM

	(a) Performance Ratio						(b) Execution time (sec)					
	Attribute Number						Attribute Number					
Query #	10	11	12	13	14	15	10	11	12	13	14	15
5	1	1	1	1	1	1.0532	0.01	0.01	0.01	0.01	0.01	0.01
6	1	1	1	1	1	1	0.01	0.02	0.02	0.02	0.02	0.02
7	1	1	1.1274	1	1	1.0007	0.02	0.01	0.02	0.01	0.03	0.03
8	1	1	1	1	1	1	0.03	0.02	0.02	0.03	0.04	0.05
9	1	1	1	1.0099	1	1	0.04	0.04	0.05	0.07	0.06	0.07
10	1	1	1	1	1	1	0.09	0.08	0.09	0.11	0.12	0.11
11	1	1.0088	1.0103	1	1	1.0003	0.11	0.11	0.14	0.13	0.12	0.21
12	1	1	1	1.0052	1	1	0.11	0.14	0.18	0.2	0.22	0.27
13	1	1	1	1	1	1	0.26	0.2	0.2	0.21	0.27	0.28
14	1	1	1	1.0041	1	1	0.21	0.26	0.37	0.46	0.42	0.43
15	1	1.0451	1	1	1.0040	1.0372	0.27	0.49	0.44	0.44	0.61	0.67

5.3 Query Level

2GM is chosen as the attribute-level optimization subroutine in our experiments of query-level optimization. The number of attributes is fixed to 15, the number of queries to 20, the maintenance bound to 10. The response time of the result generated by 2GM alone is 613431. Thus we set the response time bound from 600000 to 350000 in our experiments of testing GR and OR.

Table 3 shows that when the response time bound is lowered, the number of queries kept decreases (more queries are removed). The solutions of both GR and OR almost always keep the same number of queries. The maximum difference is only one. For efficiency of GR and OR, the execution time of GR is linear to the number of queries removed and within several minutes while that of OR is exponential to the number of queries removed.

Table 3. Table of Comparisons of Greedy Removing and Optimal Removing

RTB($\times 10^3$)		600	575	550	525	500	475	450	425	400	375	350
Queries kept	Greedy	19	19	18	18	18	17	17	16	16	16	15
	Optimal	19	19	18	18	18	17	17	17	16	16	15
Exec. time (sec)	Greedy	3.15	4.35	29.4	31.5	31.5	52.7	54.6	69.8	70.2	70.8	83.0
	Optimal	3.15	4.65	31.8	34.4	34.4	248	254	1057	1340	1504	5333

From the previous discussion, we see that the combination of 2-Greedy Merging (for attribute-level optimization) and Greedy Removing (for query-level optimization) is a good choice for solving the data cube system design problem.

6 Conclusion

In this paper, we discussed the optimization problem in requirement-based data cube system design subject to a maintenance bound and a response time bound. We proposed a two-phase approach to the problem: First, from a set of frequently asked queries, we derive an initial data cube set. Second, we remove a minimum number of cubes from the initial set to get a refined set which can satisfy the bounds. The second phase can be divided into two levels. In the query level, we remove as few data cubes from the initial set as possible. In the attribute level, we refine the resultant set to satisfy the bounds. Experiments have been done on a census database and the results show that the combination of Greedy Removing and 2-Greedy Merging is very efficient and gives a near-optimal solution. Further works include solving the data cube system design problem with considering a bound of the total storage size of data cubes. [4]

References

1. S. Chaudhuri et al. An Overview of Data Warehousing and OLAP Technology. ACM-SIGMOD Record, Vol. 26 No.1 P.65-74, March 1997
2. D.W. Cheung, B. Zhou, B. Kao, H. Lu, T.W. Lam and H.F. Ting, Requirement-based Design of Data Cube Schema. In *Proc. Eighth Int'l Conf. on Information and Knowledge Management (CIKM)*, Kanas City, Missouri, Nov 2-6 1999.
3. J. Gray, A. Bosworth, A. Layman, and H. Piramish. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *Proceeding of the 12th Intl. Conference on Data Engineering*, pages 152-159, New Orleans, February 1996.
4. E. Hung and D. W. Cheung. Optimization in Data Cube System Design. *Working paper.*
5. V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 205-216, Montreal, Quebec, June 1996.
6. P. O'Neill and G. Graefe. Multi-Table Joins Through Bitmapped Join Indexes. In *SIGMOD Record*, pages 8-11, September 1995.
7. P. O'Neil and D. Quass. Improved Query Performace with Variant Indexes. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 38-49, Tucson, Arizona, May 1997.

8. A. Shukla, P.M.Deshpande, J.F.Naughton. Materialized View Selection for Multidimensional Datasets. In *Proceedings of the International Conference on Very Large Databases*, pages 488-499, New York, USA, 1998
9. Transaction Processing Performance Council. TPC Benchmark D(Dicision Support), Standard Specification, Revision 1.2.3. San Jose, CA , USA, 1997

Exception Rule Mining with a Relative Interestingness Measure

Farhad Hussain[1], Huan Liu[1], Einoshin Suzuki[2], and Hongjun Lu[3,*]

[1] PRIS, School of Computing, National University of Singapore.
[2] Electrical and Computer Engineering, Yokohama National University, Japan.
[3] Computer Science, Hong Kong University of Science and Technology.
{farhad,liuh}@comp.nus.edu.sg, suzuki@dnj.ynu.ac.jp, luhj@cs.ust.hk

Abstract. This paper presents a method for mining exception rules based on a novel measure which estimates interestingness relative to its corresponding common sense rule and reference rule. Mining interesting rules is one of the important data mining tasks. Interesting rules bring novel knowledge that helps decision makers for advantageous actions. It is true that interestingness is a relative issue that depends on the other prior knowledge. However, this estimation can be biased due to the incomplete or inaccurate knowledge about the domain. Even if possible to estimate interestingness, it is not so trivial to judge the interestingness from a huge set of mined rules. Therefore, an automated system is required that can exploit the knowledge extractacted from the data in measuring interestingness. Since the extracted knowledge comes from the data, so it is possible to find a measure that is unbiased from the user's own belief. An unbiased measure that can estimate the interestingness of a rule with respect to the extractacted rules can be more acceptable to the user. In this work we try to show through the experiments, how our proposed relative measure can give an unbiased estimate of relative interestingness in a rule considering already mined rules.

1 Introduction

After performing data mining tasks that usually terminate with a large set of rules, there is a need to find some interesting rules that decision makers can use for advantageous actions. The number of discovered rules can, however, be so large that browsing the rule set and finding interesting rules from it can be rather difficult for the user. Moreover, it is much harder to know which of the discovered rules are really interesting. Interestingness is a relative issue since it always depends on the user's prior knowledge about the domain. However, user's belief can give a biased estimate for incomplete or inaccurate knowledge about the domain. True knowledge about the domain can be extracted form the data. Therefore, to provide an unbiased [1] measure, intuitively we can say,

* This author's work is partially supported by a grant from the National 973 project of China (No. G1998030414)

[1] In measuring interestingness user's biased belief is not incorporated

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 86–97, 2000.
© Springer-Verlag Berlin Heidelberg 2000

interestingness can be estimated relative to the common sense rules found in the data. Data mining facilitates understanding the large amount of data by generating some rules. These rules, considering as common sense knowledge, can be used to justify the interestingness in other mined rules.

We understand that something that contradicts user's common belief is bound to be interesting. Let's define exceptions as rules that contradict the common belief. Exceptions [4,8,12] can take an important role in making critical decisions. Exceptions and common sense rules point at opposite directions. Exceptions usually are minority, they are either not known (thus new) or omitted. Therefore, in many respects, for mining interesting rules, we are interested to mine exceptions for the common sense [11,12,13,14] rules. A common sense rule represents a common phenomenon that comes with high support and confidence in a particular domain. Subjectively, a common sense rule can be interesting if the user is not well aware about the problem domain or if he has a completely different view about the domain. It implies that a subjective measure of interestingness may be biased. This bias may not always be wanted in critical decision making.

Intuitively, exceptions contradict the common sense rules and they have a low support [12]. Therefore, exception rules are weak in terms of support, but having high confidence similar to those common sense rules. A weak rule of low support may not be a reliable. A user can specify minimum support for an exception to ensure mining reliable exception rules [4,12].

If we mine exception rules, literally there would be a huge number of them because of their usual lower support threshold. In this paper, an exception rule is structurally defined as shown in Table 1. Note that this is a simplified version of [11,12,13,14]. Here A and B represent a single item or a set of items.

Table 1. Rule structure for exceptions

$A \rightarrow X$	- common sense rule (strong pattern) *(high support, high confidence)*
$A, B \rightarrow \neg X$	- exception rule (weak pattern) *(low support, high confidence)*
$B \rightarrow \neg X$	-reference rule *(low support and/or low confidence)*

It is clear from the rule structure that the reference item B that explains the cause of exception, with the given common sense $A \rightarrow X$, can be of several combinations of items satisfying all the support and confidence constraints. The more the number of items we allow in the reference B, the more the candidate exception rules we can be generated for a given common sense rule. The number of reference rules also depends on the number of items for each data attribute. This implies that for a particular common sense rule, a user usually finds more and more exceptions that ultimately mislead the user for mining exceptions in

the data. Moreover, all the references that help form exceptions may not be meaningful or the user may not be aware of them due to their lower support or confidence for a particular common sense rule to be an exception. Therefore, it will be more meaningful to include those reference items about which the user is concerned. This implies that to satisfy the criteria (low support and/or low confidence) for a reference rule $B \rightarrow \neg X$ in Table 1, we simply ensure the existence of a common sense rule $B \rightarrow X$.

We are motivated to mine those exception rules that are interesting in nature and can be estimated by the knowledge from the other mined rules. Estimation of interestingness is necessary to identify the most interesting rules from the mined rules. It would be an unbiased estimate since it considers only the common sense rules in the data, not that of the user's belief. Before giving a measure of interestingness, it is necessary to understand the components of what and how they bring surprise or interestingness in a rule given the knowledge about some common sense rules.

The remainder of this paper is organized as follows. In Section 2, we talk about some preliminaries and related work on interestingness. In Section 3, we describe our approach and provide details of our measure analytically. Implementation of our algorithm for digging out the interesting rules is described in Section 4. Analysis of our experimental results are shown in Section 5. Finally, in Section 6, we present our conclusion.

2 Preliminaries and Related Work

In this paper we try to capture interestingness of a given rule through the amount of change in information relative to the common sense rules. A particular rule contains the information about its premise, predictive power and coverage. Therefore, in order to measure the relative interestingness, we calculate the difference between the information that a rule contains and the information needed to describe the given rule with common sense rules. The interestingness of the given rule is proportional to the the amount of difference between this two information. Bigger the difference in information the more interesting is the given rule. This resembles to the procedure, an expert uses to estimate the interestingness of a given rule using other rules.

We define the common sense knowledge κ for a rule, is the knowledge about some common sense rules that we can apply to estimate the interestingness in the given rule. Since, interestingness is a relative factor that depends on the knowledge about other rules, therefore, for a rule $AB \rightarrow X$, κ composed of the knowledge about the rules $A \rightarrow X$ and $B \rightarrow X$. Therefore, by our definition, the interestingness I, of a rule should be a function of its support(S), confidence(C) and the knowledge about common sense rules (κ)

$I = f(S, C, \kappa)$

We believe that lack of any one of these parameters may result in incomplete estimate of the interestingness. That means we are interested to mine those in-

teresting rules where all these three components can be applied to make it more acceptable. The measure we propose, offers both subjective and objective interestingness. When κ comes from the user's belief instead of the other mined rules, then the same measure can be considered as a subjective measure of interestingness for a given rule.

Generally, there are two categories of finding interesting rules: subjective vs. objective. In a subjective case, a user directly applies his own knowledge, sometimes even without knowing the nature of the domain to find interestingness in a rule. That is why, subjective interestingness may be biased and may vary with different users. Different methods have been proposed to capture interesting rules subjectively. The main idea is to impose user's own belief about the domain. Users usually apply their knowledge in terms of rule templates [6] and then try to match the template by scanning the data satisfying some threshold parameters. This approach may be suitable to justify a particular user's own belief system but may fail to discover some surprising rules that they even don't know. One potential problem is that user's subjective judgement may be unrealistic while applying those rules in the competitive business environment.

In an objective measure where common sense rules are not applied ($\kappa = \phi$) to estimate the interestingness, one of the main focus is to find if there is any deviation in the rule from the norm. Deviations are powerful and useful in this case as they provide a simple way of identifying interesting patterns in the data. This has been shown in the *KEFIR* [2] application. This type of measure is likely to be unbiased since no user's preference is given while estimating the interestingness. However, a rule that deviates may not be interesting if that deviation can be explained by other rules in the data. This implies that one can still generate a large number of rules that are interesting objectively but of little interest to the user [16] as they might know other rules. Again, since for the existing objective measures, $\kappa = \phi$ so, two different rules with similar support and confidence usually come with the same interestingness. Despite their similarity in support and confidence, interestingness usually should depend on the prior knowledge about those two rules. For example, using *J-measure* [10], two rules $AB \rightarrow X$ and $PQ \rightarrow X$ for which $\Pr(XAB) = \Pr(XPQ)$ and $\Pr(X|AB) = \Pr(X|PQ)$ give identical interestingness irrespective to their prior knowledge. For a rule $AB \rightarrow X$, *J-measure* is defined as follows.

$$J(X, AB) = \Pr(XAB) \log_2 \frac{\Pr(X|AB)}{\Pr(X)} + \Pr(\neg XAB) \log_2 \frac{\Pr(\neg X|AB)}{\Pr(\neg X)}$$

Though for an automated system, objective measures are always reliable due to their unbiased nature, sometimes they are completely unable to justify a rule's interestingness as they cannot handle knowledge from common sense rules. Since user's true belief will eventually build upon the common sense rules in the data, it is worth proposing a measure that can manipulate the common sense rules to estimate the interestingness in other rules. That is why, we need an objective measure that takes into account κ - the already found rules. In [11], *Geometric mean of the Average Compressed Entropies (GACE)* was defined to estimate the

interestingness of a rule if and only if the given rule is an exception to a common sense rule. *GACE* defines the interestingness in terms of *J-measure* from one rule pair (common sense and exception). For a common sense rule $A \rightarrow X$ and its exception $AB \rightarrow \neg X$, *GACE* is defined as follows

$$GACE(A \rightarrow X, AB \rightarrow \neg X) = \sqrt{J(X, A)J(\neg X, AB)}$$

When we wish to include more (such as the already extracted rule both common sense and reference) in evaluating a rule's interestingness, we need a new measure that can take into account that something more.

3 Our Approach

In this work, we are interested to mine rules that are objectively interesting, but in the meantime we are able to measure the interestingness with respect to already mined rules. For example, a rule "*Sex = F and Age = Young* $\rightarrow$ *Credit = No*" with 50% confidence may not be interesting from both subjective and objective viewpoints. From the subjective viewpoint, it is not interesting if the user believes that the females are not usually given credit compared to the males. Objectively it is also not interesting since it is not very conclusive as it has 50% confidence. However, if two strong [2] common sense rules "*Sex = F* $\rightarrow$ *Credit = Yes*" and "*Age = Young* $\rightarrow$ *Credit = Yes*" are known then "*Sex = F and Age = Young* $\rightarrow$ *Credit = No*" with 50% confidence should be interesting to us. This suggests a need to have a relative interesting measure.

We have mentioned eralier that an objective measure evaluates a rule's interestingness with its support(S) and confidence(C). Now we wish to consider the relevant extracted rules (κ) in our new measure. Below we explain in detail how all these factors together define relative interestingness of a rule.

3.1 Confidence-Based Interestingness

When no other information is given, an event with lower probability to occur gives more information, than an event with higher probability. From the information theory [9], the number of bits required to describe the occurrence is defined as

I = - $\log_2 P$
where,
P = The probability of that event to occur.

Similarly, for a given rule $AB \rightarrow X$ with confidence $\Pr(X|AB)$, will require $-\log_2(\Pr(X|AB))$ and $-\log_2(\Pr(\neg X|AB))$ number of bits to describe the events X and $\neg X$ given AB. Note that in the case of analyzing the rules we not only consider the event X or $\neg X$ but also the premise that causes the events to occur. That is why in the case of rules, whenever we talk about the events X or

[2] High support and high confidence

$\neg X$, we always mention the premise which leads the events to occur. However, in information theory, we are not interested to know how the events X and $\neg X$ occur. Since the probability of occurrence of the events X and $\neg X$ given AB are $\Pr(X|AB)$ and $\Pr(\neg X|AB)$, so, the expected number of bits required to describe events X and $\neg X$ influenced by AB are $-\Pr(X|AB)\log_2(\Pr(X|AB))$ and $-\Pr(\neg X|AB)\log_2(\Pr(\neg X|AB))$ respectively. Thus the total number of bits required to describe the rule $AB \rightarrow X$ is

$$I_c^{AB_0} = (-\Pr(X|AB)\log_2\Pr(X|AB)) + (-\Pr(\neg X|AB)\log_2\Pr(\neg X|AB))$$

where,

$I_c^{AB_0}$ = Number of bits required to describe the rule $AB \rightarrow X$ when no other knowledge have been applied.

However, the difference in number of bits in describing the rule $AB \rightarrow X$ in terms of $A \rightarrow X$ and $B \rightarrow X$ can bring surprise. Bigger the difference in describing the rule $AB \rightarrow X$, the more it is interesting. Therefore, to estimate the relative interestingness in terms of rules $A \rightarrow X$ and $B \rightarrow X$, we need to know the number of bits required to describe the event X when the probability of that event to occur given A and B are $\Pr(X|A)$ and $\Pr(X|B)$ respectively. The Table 2 shows the number of bits required to describe the events X and $\neg X$ when A and B are given.

Table 2. The events and the information

Events	Given	Number of bits
X	A	$-\log_2\Pr(X\|A)$
¬X	A	$-\log_2\Pr(\neg X\|A)$
X	B	$-\log_2\Pr(X\|B)$
¬ X	B	$-\log_2\Pr(\neg X\|B)$

Since the rule $AB \rightarrow X$ describes the event X in terms of A and B so, to describe the similar event X, in terms of A and [3] B using the rules $A \rightarrow X$ and $B \rightarrow X$ we need $-\log_2(\Pr(X|A))$ and $-\log_2(\Pr(X|B))$ number of bits. Now, in rule $AB \rightarrow X$ probability of the event X to occur is $\Pr(X|AB)$. Therefore, the expected number of bits required to describe all the X events in rule $AB \rightarrow X$ in terms of A and B using the two rules is thus $-\Pr(X|AB)(\log_2(\Pr(X|A)) + \log_2(\Pr(X|B)))$. Similarly, for the event $\neg X$ in rule $AB \rightarrow X$, $-\Pr(\neg X|AB)(\log_2(\Pr(\neg X|A)) + \log_2(\Pr(\neg X|B)))$ number of bits will be required. Thus the total number of bits required to describe the event X and $\neg X$ in the rule $AB \rightarrow X$ by rules $A \rightarrow X$ and $B \rightarrow X$ is

$$I_c^{AB_1} = -\Pr(X|AB)[\log_2\Pr(X|A) + \log_2\Pr(X|B)] - \Pr(\neg X|AB)[\log_2\Pr(\neg X|A) + \log_2\Pr(\neg X|B)]$$

where,

[3] It is not **or** because, event X was influenced in rule $AB \rightarrow X$ by A and B

$I_c^{AB_1}$ = Number of bits required when $AB \rightarrow X$ is described by $A \rightarrow X$ and $B \rightarrow X$.

Thus, the relative surprise or relative interestingness that comes from the difference between two descriptions for the given rule $AB \rightarrow X$ is

$$\begin{aligned} RI_c^{AB} &= I_c^{AB_1} - I_c^{AB_0} \\ &= -\Pr(X|AB)\log_2 \frac{\Pr(X|A)\Pr(X|B)}{\Pr(X|AB)} - \Pr(\neg X|AB)\log_2 \frac{\Pr(\neg X|A)\Pr(\neg X|B)}{\Pr(\neg X|AB)} \\ &= \Pr(X|AB)\log_2 \frac{\Pr(X|AB)}{\Pr(X|A)\Pr(X|B)} + \Pr(\neg X|AB)\log_2 \frac{\Pr(\neg X|AB)}{\Pr(\neg X|A)\Pr(\neg X|B)} \end{aligned}$$

where,

RI_c^{AB} = The relative surprise or interestingness of the rule, considering the confidence and the knowledge about other rules.

The interestingness of a rule that we have formulated in terms of the confidence gives the exact impression of relative entropy [15]. Here the entropy of a rule is calculated relative to the other rules. It is a measure of distance between two distributions. In statistics, it arises as an expected logarithm of the likelihood ratio. The relative entropy $D(p(x)||q(x))$ is a measure of the inefficiency of assuming that the distribution is $q(x)$ when the true distribution is $p(x)$. The relative entropy or *Kullback Leibler distance* between two probability function is defined as,

$$D(p(x)||q(x)) = \sum_{x \in X} p(x) log \frac{p(x)}{q(x)}$$

In estimating the interestingness of the rule $AB \rightarrow X$ with true confidence $\Pr(X|AB)$ we approximated its confidence from the rules $A \rightarrow X$ and $B \rightarrow X$.

3.2 Support-Based Interestingness

By support of a rule $AB \rightarrow X$, we mean the frequency of the rule's consequent evaluated as X by AB relative to the whole dataset. When we know the support of two common sense rules $A \rightarrow X$ and $B \rightarrow X$, we know the relative frequency of the consequent X and $\neg X$ evaluated by A and B separately. A similar relative entropy measure can be applied to estimate the surprise from support. Now, for the newly discovered rule $AB \rightarrow X$, the true distribution of the consequent X and $\neg X$ evaluated by A and B are $\Pr(ABX)$ and $\Pr(AB\neg X)$ respectively. From the knowledge of our one common sense rule $A \rightarrow X$, for which the relative frequency of X and $\neg X$ are $\Pr(AX)$ and $\Pr(A\neg X)$ respectively, can be used to find the distance between two distributions of consequent using relative entropy. The relative entropy of $AB \rightarrow X$ relative to the rule $A \rightarrow X$ in terms of their support is thus

$$D(AB \rightarrow X||A \rightarrow X) = \Pr(ABX)\log \frac{\Pr(ABX)}{\Pr(AX)} + \Pr(AB\neg X)\log \frac{\Pr(AB\neg X)}{\Pr(A\neg X)}$$

Similarly for rule $B \rightarrow X$, the relative entropy is

$$D(AB \rightarrow X||B \rightarrow X) = \Pr(ABX)\log \frac{\Pr(ABX)}{\Pr(BX)} + \Pr(AB\neg X)\log \frac{\Pr(AB\neg X)}{\Pr(B\neg X)}$$

Thus the total relative interestingness due to rule's support that comes from the relative entropy of $AB \rightarrow X$ for the two common sense rule is,

$$RI_s^{AB} = D(AB \rightarrow X||A \rightarrow X) + D(AB \rightarrow X||B \rightarrow X)$$
$$= \Pr(ABX) \log \frac{\Pr(ABX)}{\Pr(AX)\Pr(BX)} + \Pr(AB\neg X) \log \frac{\Pr(AB\neg X)}{\Pr(A\neg X)\Pr(B\neg X)}$$

Hence, the total interestingness of a rule $AB \rightarrow X$ relative to $A \rightarrow X$ and $B \rightarrow X$ is

$RI = RI_c^{AB} + RI_s^{AB}$

This includes support, confidence and consideration of other rules to estimate the relative surpriseness.

4 Digging out the Exceptions

Since exceptions are weak in terms of support, we are supposed to dig deeper into the data with lower support threshold to bring them out. Applying lower support threshold for mining exceptions is not a cost-effective solution. Moreover, in that case, a large number of rules will be generated where not all of them are exceptions. Actually we are going to mine those exceptions where the rules extracted as a common sense will be used to alleviate the problem of dealing with lower support threshold. In other words, we search for reliable exceptions starting from the common sense rules. To satisfy all the constraints defined in Table 1, we find exception $AB \rightarrow \neg X$ from two common sense rules $A \rightarrow X$ and $B \rightarrow X$ ($B \rightarrow X$ as common sense infers $B \rightarrow \neg X$ to be reference for its obvious low support or/and low confidence). By doing this we can estimate the amount of surprise the exception rule brings from the knowledge of the extracted rules. From Figure 1 we can visualize how exceptions are mined going deeper into the data. The threshold *CS support* is the minimum support to mine the common sense rules from the data and the *EX support* to assure the reliability of the exception rules. The following algorithm describes the way we mine interesting rules.

begin
$LI = \phi$ // list containing large item set
$LC = \phi$ // list containing common sense rules
$LR = \phi$ // list containing reference rules for a common sense
$LE = \phi$ // list containing candidate exception rules
$LI \leftarrow$ *GenerateLargeItemSet()* // running apriori [1]
$LC \leftarrow$ *GenerateAllCommonSense(LI)*
for each CS_i **from** LC **do**
  $A \leftarrow$ *GetAntecedent*(CS_i)
  GetReferences(CS_i, LC)
  for each RR_j **from** LR **do**
    $B \leftarrow$ *GetAntecedent*(RR_j)
    if $(A \cup B)$ **is not in** LI
      insert($A \cup B \cup \neg$*Consequent*(CS_i), LE)

```
    end for
end for
LE ← GenerateExceptions(LE)              // Database scan once
EstimateInterestingness(LC, LE)          // Output interesting rules
end.                                     // according to the degree of surprise
```

The function *GetReferences*(CS_i, *LC*), returns all the candidate reference rules for CS_i, from *LC*. The reference rules are those common sense rules in *LC* that have similar consequent as that of CS_i. Once we have inserted all the candidate exception rules into *LE*, we scan the database once to obtain the support and confidence of each candidate exceptions. We output those rules that satisfy the thresholds using *GenerateExceptions(LE)*. *EstimateInterestingness(LC, LE)* estimates the relative interestingness.

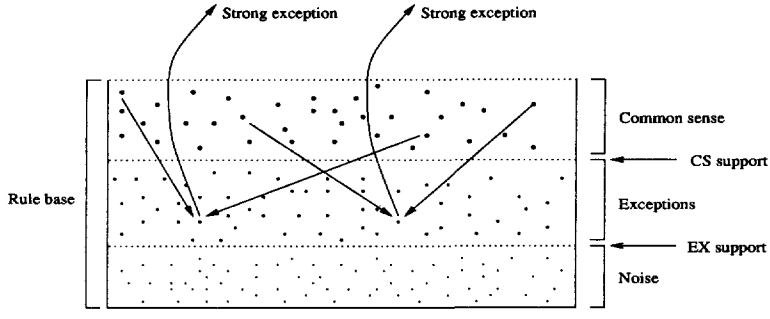


Fig. 1. Rules in the data

5 Experiments

In this section we explain our interesting rules obtained from Japanese credit data and mushroom data [7]. The credit data set has 10 attributes and 125 instances with a binary class. The two types of classes define when a credit is given to a particular person depending on other attribute values. Based on our approach we obtain the strong patterns that would eventually be considered as a common sense rule for that credit domain. We use 20% support threshold for common sense. To mine the reliable exceptions considering already extracted rules, we use 5% support. For both cases we mine those having more than or equal to 50% confidence.

We will show the effects of the proposed measure (RI) in two settings: (1) removing reference rules by assuming they are not available; and (2) including reference rules. As in the first case, GACE has been shown effective. Therefore we compare RI with GACE to see if RI can achieve the planned objective: they

should identify the interesting exceptions as GACE does. In the second case, we show that the ranking order changes due to considering reference rules.

Table 3. Exception rules (credit) justified by the common sense rules

Common Sense (CS) Reference Rule (RR) Exception = CS + RR	Credit Given	Conf %	Supp %	RI With Ref.	RI Without Ref.	GACE
Sex = F, MPay = 0-20 (CS)	Yes	61	26			
Save = 0-100, Mnts = 0-10 (RR)	No	37	17			
Sex = F, MPay = 0-20, Save = 0-100, Mnts = 0-10	No	54	11	3.16	1.98	0.013
MPay = 0-20, Mnts = 0-10 (CS)	Yes	65	32			
Sex = F (RR)	No	38	17			
Sex = F, MPay = 0-20, Mnts = 0-10	No	52	10	3.00	1.93	0.006
Mnts = 0-10 (CS)	Yes	65	32			
Sex = F, Save = 0-100 (RR)	No	42	18			
Sex = F, Save = 0-100, Mnts = 0-10	No	52	10	2.96	1.93	0.006
Sex = F, Age = 0-40 (CS)	Yes	61	20			
Marry = Yes (RR)	No	29	15			
Sex = F, Age = 0-40, Marry = Yes	No	51	6	2.52	1.67	0.007

From the experimental result it is clear that RI without considering reference rules behaves similar as GACE. So, RI can identify reliable exceptions. When considering reference rules, RI has changed the rankings (2nd and 3rd rule sets in Table 3). An *actionable* exception rule should be reliable. An exception would be more reliable if it has higher confidence and support. Therefore, from the perspective of both interestingness and reliability, the first exception "*Sex = F, MPay = 0-20, Save = 0-100, Mnts = 0.10 → Credit = No*" should have the highest ranking among the four exception rules.

For the mushroom data (22 attributes and 8124 tuples) we conduct a similar experiment with 15% support and 55% confidence for mining common sense rules. To mine the exceptions, we specify 5% support and 55% confidence. Table 4 shows some of the rules and their corresponding interestingness, when RI without reference cannot diferentiate the difference between the third and four rule sets, RI with reference can. Hence, when we have some knowledge about the rule, the knowledge should be used if we wish to find relative interestingness. GACE has made use of common sense rules and RI goes one step further to include reference rules in the measure.

6 Conclusion

In this work we define an objective measure of relative interestingness that ties up the with common sense and reference rules in the data when estimating interestingness. This opens the door to apply RI to measure subjective interestingness

Table 4. Exception rules (mushroom) justified by the common sense rules

Common Sense (CS) Reference Rule (RR) Exception = CS + RR	Edible/ Poisonous	Conf %	Supp %	RI With Ref.	RI Without Ref.	GACE
stalk-root = ? (CS)	P	71	22			
bruises = f, gill-size = b,						
stalk-shape = e, veil-type = p (RR)	E	27	6			
CS + RR	E	100	6	6.76	3.98	0.05
ring-type = e (CS)	P	64	22			
bruises = f, gill-size = b,						
ring-number = t (RR)	E	43	12			
CS + RR	E	100	9	4.71	3.58	0.04
gill-color = p, veil-type = p,						
veil-color = w (CS)	P	57	23			
stalk-root = ? (RR)	E	29	9			
CS + RR	E	94	6	4.19	2.28	0.02
gill-attachment = f, stalk-shape = e,						
veil-type = p (CS)	P	57	23			
cap-surface = s, veil-color = w (RR)	E	40	11			
CS + RR	E	94	6	3.31	2.28	0.02

if we can describe an expert's knowledge in rules. For example, we wish to evaluate the new rule in Table 5; it is not very interesting than those in Table 3. This should be the case as the new rule is not an exception.

Table 5. A new rule (credit) justified by the common sense rules

Common sense Common sense New rule	Credit Given	Conf %	Supp %	RI
Jobless = No, Sex = F	Yes	70	28	
Married = No	Yes	64	30	
Jobless = No, Sex = F, Married = No (Not exception)	Yes	71	12	2.34

In our approach we focus on relative interestingness that evaluates an exception rule with respect to its common sense and reference rules. Not much extra costs invloved in calculating RI: Using Apriori [1] to find the frequent itemsets, then we just need another database scan to estimate the support and confidence of the candidate exception items. This efficient new measure provides one more means in exception rule mining along with other interestingness measures.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th conference on Very Large Databases (VLDB)*, pages 478–499, 1994.
2. G. Piatetsky-Shapiro C. Matheus and D. McNeil. *Selecting and Reporting What is Interesting: The KEFIR Application to Healthcare Data. AAAI Press/ MIT Press*, 1996.
3. Sarawagi Chakrabarti. Mining surprising patterns using temporal description length. In *Proc. 24th on Very Large Databases (VLDB)*, pages 606–616, 1998.
4. Liu H. and Lu H. Efficient search of reliable exceptions. In *Proc. third Pacific-Asia conference on Knowledge Discovery and Data mining (PAKDD)*, pages 194–203, 1999.
5. W. Hsu Liu B. and Shu Chen. Using general impression to analyze discovered classification rules. In *Proc. third international conference on Knowledge Discovery and Data mining (KDD)*, pages 31–36, 1997.
6. H. Mannila M. Klemettinen. Finding interesting rules from large sets of discovered association rules. In *Third Intl. Conference on Information and Knowledge Management (CIKM)*, 1994.
7. C.J. Merz and P.M. Murphy. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA: University of California, Department of Information and Computer Science, 1996.
8. B. Padmanabhan and A. Tuzhilin. A beleif-driven method for discovering unexpected patterns. In *Proc. fourth international conference on Knowledge Discovery and Data mining (KDD)*, pages 27–31, 1998.
9. C. Shannon and W. Weaver. *The Mathematical Theory of Information.* Urbana: University of Illinois Press, 1949.
10. P. Smyth and Goodman R. M. Rule induction using information theory. In *Knowledge Discovery in Databases, Piatetsky-Shapiro, G. AAAI Press / The MIT Pres*, pages 159–176, 1991.
11. E. Suzuki. Discovering unexpected exceptions: A stochastic approach. In *Proc. RFID*, pages 225–232, 1996.
12. E. Suzuki. Autonomous discovery of reliable exception rules. In *Proc. third international conference on Knowledge Discovery and Data mining (KDD)*, pages 259–262, 1997.
13. E. Suzuki and Y. Kodratoff. Discovery of surprising exception rules based on intensity of implication. In *Proc. second Pacific-Asia conference on Knowledge Discovery and Data mining (PAKDD)*, 1998.
14. E. Suzuki and M. Shimura. Exceptional knowledge discovery in databases based on information theory. In *Proc. second international conference on Knowledge Discovery and Data mining (KDD)*, pages 295–298, 1996.
15. C. Thomas M and J. Thomas A. *Elements of Information Theory.* Wiley-Interscience Publication, 1996.
16. A. Tuzhilin and A. Silberschatz. What makes patterns interesting in knowledge discovery systems. In *IEEE Trans. Knowledge Discovery and Data Engineering*, pages 970–974, 1996.

Consistency Based Feature Selection

Manoranjan Dash[1], Huan Liu[1], and Hiroshi Motoda[2]

[1] School of Computing, National University of Singapore, Singapore.
[2] Division of Intelligent Sys Sci, Osaka University, Ibaraki, Osaka 567, Japan.

Abstract. Feature selection is an effective technique in dealing with dimensionality reduction for classification task, a main component of data mining. It searches for an "optimal" subset of features. The search strategies under consideration are one of the three: complete, heuristic, and probabilistic. Existing algorithms adopt various measures to evaluate the goodness of feature subsets. This work focuses on one measure called *consistency*. We study its properties in comparison with other major measures and different ways of using this measure in search of feature subsets. We conduct an empirical study to examine the pros and cons of these different search methods using consistency. Through this extensive exercise, we aim to provide a comprehensive view of this measure and its relations with other measures and a guideline of the use of this measure with different search strategies facing a new application.

1 Introduction

Classification is an important data mining task. The basic problem of classification is to classify a given pattern (example) to one of m known classes. A pattern of features presumably contains enough information to distinguish among the classes. When a classification problem is defined by features, the number of features (N) can be quite large. Pattern classification is inherently connected to information reduction. Features can also be redundant or irrelevant. An irrelevant feature does not affect the underlying structure of the data in any way. A redundant feature does not provide anything new in describing the underlying structure. Because redundant and irrelevant information is cached inside the totality of the features, a classifier that uses all features will perform worse than a classifier that uses relevant features that maximize interclass differences and minimize intraclass differences [4]. Feature selection is a task of searching for "optimal" subset of features from all available features. Its motivation is three-fold: *simplifying* the classifier; *improving* the accuracy of the classifier; and *reducing* data dimensionality for the classifier. The last point is particularly relevant when a classifier is unable to handle large volumes of data.

Features may not be all relevant. In order to measure the usefulness (or goodness) of selected features, we need selection criteria. The class separability is often used as one of the basic selection criteria. When a set of features maximizes the class separability, it is considered well suited for classification. From a statistics viewpoint, five different measurements for class separability

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 98-109, 2000.
© Springer-Verlag Berlin Heidelberg 2000

are analyzed in [8]: error probability, interclass distance, probabilistic distance, probabilistic dependence and entropy. Information-theoretic considerations [20] suggested something similar: using a good feature of discrimination provides compact descriptions of each class, and these descriptions are maximally distinct. Geometrically, this constraint can be interpreted to mean that (i) such a feature takes on nearly identical values for all examples of the same class, and (ii) it takes on some different values for all examples of the other class. In this work, we use a selection criterion that does not attempt to maximize the class separability but tries to retain the discriminating power of the data defined by original features. Feature selection is formalized as finding the smallest set of features that can distinguish classes as if with the full set. In other words, with a subset S of features, no two examples with the same values on S have different class labels [1]. We study the pros and cons of this measure in comparison with other measures. Another aspect of feature selection is related to the study of search strategies. Extensive research efforts have been devoted to this study [19, 7, 3]. Examples are Branch &Bound [16], Relief [11], Wrapper methods [12], and Las Vegas algorithms [14]. The search process starts with either an empty set or a full set. For the former, it expands the search space by adding one feature at a time (Forward Selection) - an example is Focus [1]; for the latter, it expands the search space by deleting one feature at a time (Backward Selection) - an example is 'Branch & Bound' [16].

The contributions of this paper are: (a) studying a monotonic criterion for feature selection w.r.t. other selection criteria; (b) exploring its properties and use in exhaustive (complete), heuristic, and probabilist search; (c) comparing its different uses with a number of data sets; and (d) suggesting a framework of when to use what. In the rest of the paper P is the number of patterns, N is the number of features, M is the size of relevant features, and m is the number of class labels.

2 Consistency Measure

Consistency can be interpreted as zero inconsistency. If we attain zero inconsistency, we achieve 100% consistency. Throughout this paper we use consistency and inconsistency interchangeably.

2.1 The Measure

The suggested measure U is an *inconsistency rate* over the data set for a given feature set. In the following description *pattern* means a set of values for the features in a candidate subset. The inconsistency rate is calculated as follows: (1) two patterns are considered inconsistent if they match all but their class labels, for example, an inconsistency is caused by two instances (0 1 a) and (0 1 $\bar{a}$) with different classes (a and $\bar{a}$); and (2) the inconsistency count for a pattern is the number of times it appears in the data minus the largest number among different class labels: for example, let us assume there are n matching patterns,

among which c_1 patterns belong to label_1, c_2 to label_2, and c_3 to label_3 where $c_1+c_2+c_3 = n$. If c_3 is the largest among the three, the inconsistency count is $(n-c_3)$; and (3) the inconsistency rate is the sum of all the inconsistency counts for all possible patterns of a feature subset divided by the total number of patterns (P). By employing a hashing mechanism, we can compute the inconsistency rate approximately with a time complexity of $O(P)$. Unlike the commonly used univariate measures in literature [18], this is a multivariate measure which checks a subset of features at a time.

2.2 Other Evaluation Measures

An optimal subset is always relative to a certain evaluation function. An optimal subset chosen using one evaluation function may not be the same as that using another evaluation function. Typically, an evaluation function tries to measure the discriminating ability of a feature or a subset to distinguish the different class labels. Blum and Langley [3] grouped different feature selection methods into two broad groups (i.e., filter and wrapper) based on their use of an inductive algorithm in feature selection or not. *Filter* methods are independent of an inductive algorithm, whereas *wrapper* methods are not. Ben-Bassat [2] grouped the evaluation functions until 1982 into three categories: information or uncertainty measures, distance measures, and dependence measures. Considering these divisions and latest developments, we divide the evaluation functions into five categories: *distance, information (or uncertainty), dependence, consistency,* and *classifier error rate.* **Distance Measures** It is also known as separability, divergence, or discrimination measure. For a two-class problem, a feature X is preferred to another feature Y if X induces a greater difference between the two-class conditional probabilities than Y; if the difference is zero then X and Y are indistinguishable. An example is Euclidean distance. **Information Measures** These measures typically determine the information gain from a feature. The information gain from a feature X is defined as the difference between the prior uncertainty and expected posterior uncertainty using X. Feature X is preferred to feature Y if the information gain from feature X is greater than that from feature Y [2]. An example is entropy. **Dependence Measures** Dependence measures or correlation measures quantify the ability to predict the value of one variable from the value of another variable. Correlation coefficient is a classical dependence measure and can be used to find the correlation between a feature and a class. If the correlation of feature X with class C is higher than the correlation of feature Y with C, then feature X is preferred to Y. A slight variation of this is to determine the dependence of a feature on other features; this value indicates the degree of redundancy of the feature. All evaluation functions based on dependence measures can be divided between distance and information measures. But, these are still kept as a separate category because, conceptually, they represent a different viewpoint [2]. **Consistency Measures** This type of measures has been in focus recently. They are characteristically different from other measures because of their heavy reliance on the training data and use of Min-Features bias in selecting a subset of features [1]. Min-Features bias prefers

consistent hypotheses definable over features as few as possible. This measure is similar to the consistency measure U we described in the beginning of this section with the difference that U can handle noise (e.g. misclassification). **Error Rate Measures** The methods using this type of evaluation function are called "wrapper methods", *i.e.*, the classifier is the evaluation function. As the features are selected using the classifier that later uses these selected features in predicting the class labels of unseen instances, the accuracy level is very high although computationally rather costly [9].

2.3 Consistency Measure vis-a-vis Other Measures

We compare consistency measure with other measures. First, consistency measure is monotonic and others are not. Assuming we have subsets $\{S_0, S_1, ..., S_n\}$ of features, we have a measure U that evaluates each subset S_i. The monotonicity condition requires the following: $S_0 \supset S_1 \supset ... \supset S_n \Rightarrow U(S_0) \leq U(S_1) \leq ... \leq U(S_n)$. Second, for the consistency measure, a feature subset can be evaluated in $O(P)$. It is usually costlier for other measures. For example, to construct a decision tree in order to have predictive accuracy, it requires at least $O(P \log P)$; to calculate the distances, it requires $O(P^2)$. Third, consistency measure can help remove both redundant and irrelevant features; other measures may not do so. Last, consistency measure is capable of handling some noise in the data reflected as a percentage of inconsistencies. This percentage can be obtained by going through the data once. In short, consistency measure is monotonic, fast, able to remove redundant and/or irrelevant features, and capable of handling some noise[1]

3 Ways of Using Consistency Measure

Different search strategies pose further constraints on a selection criterion. We demonstrate that the consistency measure can be employed in common forms of search without modification. Five different algorithms represent standard search strategies: *exhaustive* - Focus [1], *complete* - ABB [13], *heuristic* - SetCover [6], *probabilistic* - LVF [14], and *hybrid* of ABB and LVF - QBB. We examine their advantages and disadvantages.

Focus: exhaustive search: Focus [1] starts with an empty set and carries out breadth-first search until it finds a minimal subset that predicts pure classes. With some modification of Focus, we have FocusM that can work on non-binary data with noise. As FocusM is exhaustive search it guarantees an optimal solution. However, FocusM's time performance can deteriorate if M is not small with respect to N. The search space of FocusM is closely related to the number of relevant features. In general, the less the number of relevant features, the smaller the search space.

ABB: complete search: Branch & Bound (B&B) [16] starts with a full set

[1] There are many types of noise. Consistency measure can handle misclassifications.

of features, and removes one feature at a time. When there is no restriction on expanding nodes in the search space, this could lead to an exhaustive search. However, if each node is evaluated by a measure U and an upper limit is set for the acceptable values of U, then B&B backtracks whenever an infeasible node is discovered. If U is monotonic, no feasible node is omitted and savings of search time do not sacrifice optimality. As pointed out in [19], the measures used in [16] such as accuracy have disadvantages (e.g., non-monotonicity); the authors of [19] proposed the concept of approximate monotonicity. ABB [13] is an automated B&B algorithm having its bound as the inconsistency rate of the data when the full set of features is used. It starts with the full set of features S^0, removes one feature from S_j^{l-1} in turn to generate subsets S_j^l where l is the current level and j specifies different subsets at the lth level. If $U(S_j^l) > U(S_j^{l-1})$, S_j^l stops growing (its branch is pruned); otherwise, it grows to level $l+1$, *i.e.* one more feature could be removed.

Since inconsistency is a monotonic measure, ABB guarantees an optimal solution. However, a brief analysis suggests that ABB's time performance can deteriorate as the difference $N - M$ increases. This issue is related to how many nodes (subsets) have been generated. The search space of ABB is closely related to the number of relevant features. In general, the more the number of relevant features, the smaller the search space due to early pruning of the illegitimate nodes. Our analysis of Focus and ABB reveals that Focus is efficient when M is small, and ABB is efficient when $N - M$ is small. In other cases, we can use inconsistency measure in heuristic search.

SetCover: heuristic search: SetCover [6] uses the observation that the problem of finding the smallest set of consistent features is equivalent to 'covering' each pair of examples that have different class labels with some feature on which they have different values. This enables us to apply Johnson's algorithm [10] for Set Cover for this problem, which implies that the resulting algorithm outputs a consistent feature set of size $O(M \log P)$ in polynomial time. Variants of Set Cover have previously been used for learning conjunctions of boolean features. Consistency criterion can be restated as: a feature set S is consistent if for any pair of instances with different class labels, there is a feature in S that takes different values. Thus including a feature f in S "takes care of" all those example pairs with different class labels on which f takes different values. Once all pairs are "taken care of" the resulting set S is consistent. The advantages of SetCover is that it is fast, close to optimal, and deterministic. This works well for data where features are rather independent of each other. It may, however, have problems where features have inter-dependencies. This is because it selects the best feature in each iteration based on the number of instance-pairs covered. A new solution is needed that avoids the problems of exhaustive and heuristic search. Probabilistic search is a natural choice.

LVF: probabilistic search: Las Vegas algorithms [5] for feature subset selection can make probabilistic choices of subsets in search of an optimal set. Another similar type of algorithms is the Monte Carlo algorithm in which it is often possible to reduce the error probability arbitrarily at the cost of a slight increase in

computing time [5]. LVF is more suitable since the probability of generating a certain subset is the same. LVF adopts the inconsistency rate as the evaluation measure. Due to its monotonicity, a superset of a subset of relevant features is also good. Hence, there are more chances for good subsets to be selected. LVF keeps the smallest subset of features randomly generated so far that satisfies a threshold (by default it is the inconsistency rate of the data with all features). It is fast in reducing the number of features. We conducted experiments to observe how the number of valid features (M') drops as the number of randomly generated feature sets increases. A total of 10 data, both artificial and real, are chosen for the experiments from the UC Irvine data repository [15] (Table 1). Two typical graphs are shown in Figure 1 in a longer time span (partial results shown in Table 1) in order to observe the trend.

Data	LED24	Lung	Lymph	Mush	Par3+3	Promo	Soy	Splice	Vote	Zoo
P	1200	32	148	7125	64	106	47	3190	435	74
m	10	3	4	2	2	2	4	3	2	7
N	24	56	18	22	12	57	35	60	16	16
$M'(M)$	12(5)	19(4)	8(6)	8(4)	5(3)	15(4)	12(2)	19(9)	13(8)	9(5)
#Eval	230	155	215	22	25	187	42	284	215	25
#Max	2^{24}	2^{56}	2^{18}	2^{22}	2^{12}	2^{57}	2^{35}	2^{60}	2^{16}	2^{16}

Table 1. The number of valid features (M') drops sharply in the first few hundred runs for all data. P, N, M and m are defined earlier. #Eval is number of subsets generated and evaluated. #Max is maximum possible subsets.

The trend found in all the experiments is that M' drops sharply from N in the first few hundred runs (one run means one feature set is randomly generated and evaluated). Afterwards, it takes quite a long time to further decrease M'. Some analysis can confirm this finding. A particular set has a probability of $1/2^N$ to be generated. At the beginning, the full set is valid. Many subsets can satisfy the inconsistency criterion. As M' decreases from N to M, fewer and fewer subsets can satisfy the criterion. However, the probability of a distinct set being generated is still $1/2^N$. That explains why the curves have a sharp drop in the beginning and then become flat in Figure 1. LVF reduces the number of features quickly during the initial stage (the first few hundred loops); after that LVF still searches in the same way (i.e., blindly), the computing resource is spent on generating many subsets that are obviously not good.

QBB: hybrid search: As ABB and LVF complement each other, QBB is a natural offspring of ABB and LVF, which uses inconsistency as its evaluation measure. QBB runs LVF in the first phase and ABB in the second phase so that the search is more focused after the sharp decrease in the number of valid subsets. A key issue remains: what is the crossing point in QBB at which ABB takes over from LVF. If we allow only certain amount of time to run QBB, the point at which ABB takes over from LVF is crucial for the efficiency of QBB.

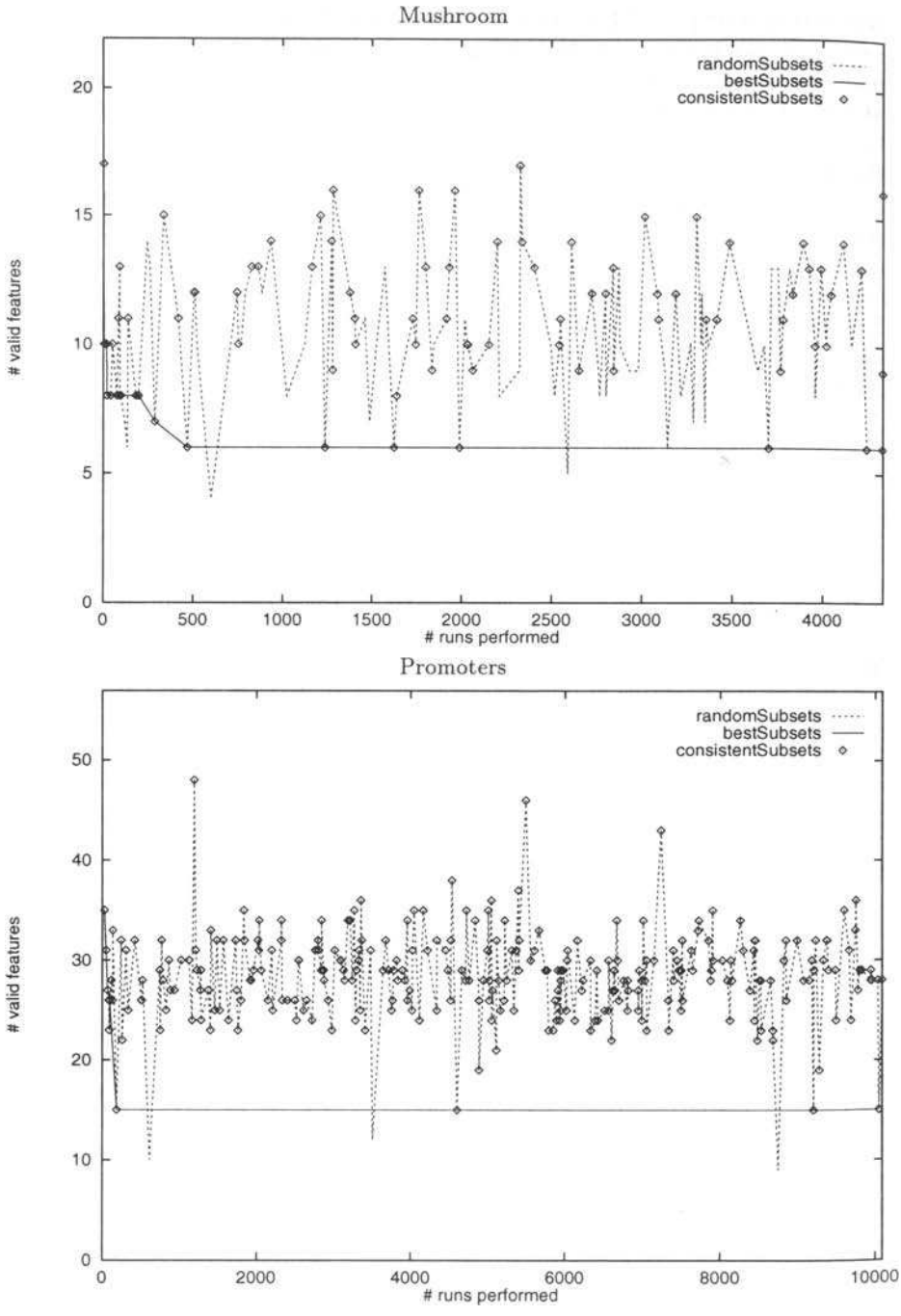


Fig. 1. The typical trends of the decreasing number of valid features versus the number of runs performed. Points include both valid and invalid feature subsets. Valid subsets are connected by solid lines.

Extensive experiments suggested that dividing the total time equally between LVF and ABB is a robust solution and is more likely to yield the best results. If the crossing point is too early, LVF might not have reduced the valid subset size substantially for ABB to perform well under time constraint; but if the crossing point is too late, the small sized subsets generated by LVF at the crossing point might not contain any minimal size subset, and so ABB becomes ineffective.

3.1 Summary: When to Use What

As we have five algorithms to choose from, we are also interested to know how we should use them. Theoretical analysis and experimental experience suggest the following. If M - the size of relevant features is small, FocusM should be chosen; however if M is even moderately large, FocusM will take a long time. If there are a small number of irrelevant and redundant features, ABB should be chosen; but ABB will take a long time for a moderate number of irrelevant features. For data with large numbers of features, FocusM and ABB should not be expected to terminate in realistic time. For the Letter data with 20,000 instances ($N = 16$ and $M = 11$) FocusM took more than 2 days to terminate whereas ABB took more than 7 hours to generate optimal subsets. Hence, in such cases one should resort to heuristic or probabilistic search for faster results. Although these algorithms may not guarantee optimal subsets but will be efficient in generating near optimal subsets in much less time. SetCover is heuristic, fast, and deterministic. It may face problems with data having highly interdependent features. LVF is probabilistic, not prone to the problem faced by SetCover, but slow to converge in later stages. As we have shown, it can reduce the feature subset size very fast in the beginning but then it slows down in reducing features. QBB is a welcome modification as it captures the best of LVF and ABB. It is reasonably fast (slower than SetCover), robust, and can handle features with high interdependency.

4 Further Experiments

The points that remain inconclusive are: (1) features selected using inconsistency can achieve the objective of dimensionality reduction without sacrificing predictive accuracy; and (2) how the different algorithms fare in terms of time and optimality. The experimental procedure is to (1) choose data frequently used by the community; (2) run ABB to get the minimal size as reference; (3) compare the performance (average time and number of selected features) of different algorithms; and (4) compare the accuracy of two different classifiers (C4.5 [17] and Back-propagation neural network [21]) over data before and after feature selection by QBB. Ten data, both artificial and real, are chosen for the experiments from the UC Irvine data repository [15]. A summary of these data is given in Table 1. Par3+3 contains 12 features (3 relevant, 3 redundant, 6 irrelevant).

Figure 2 shows a comparison of the performance (both average time and number of selected features) of different algorithms. First ABB is run over the 10 data to find the M (minimal size) values. For comparison purpose we have

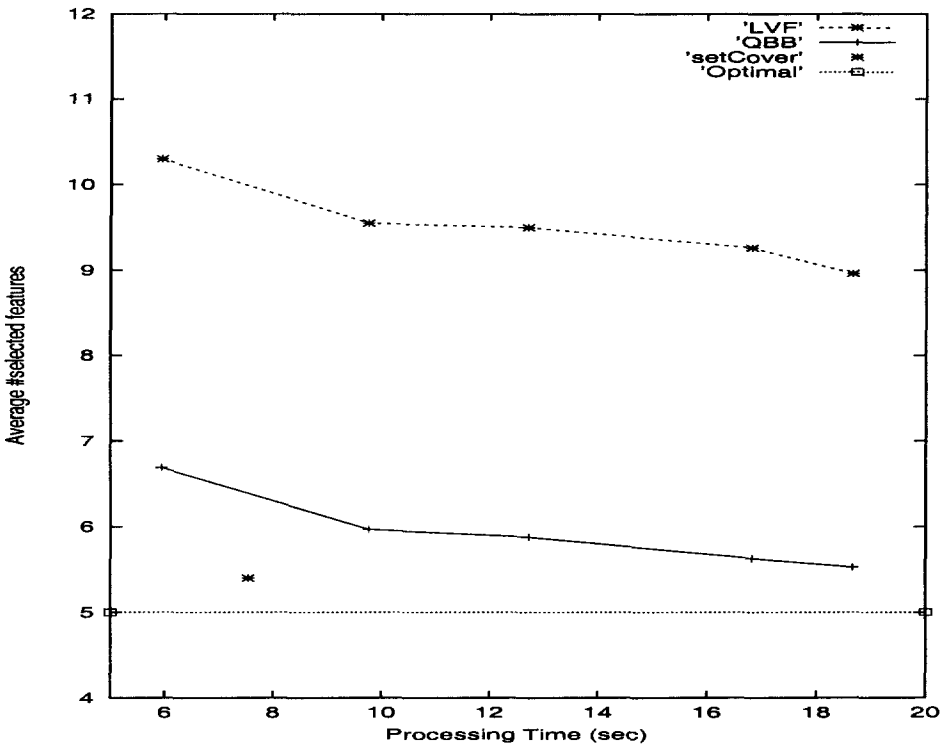


Fig. 2. Experiments to show how differently algorithms fare in terms of time and optimality. Results of *Focus* and *ABB* are out of bounds in x-axis (time).

calculated the average minimal value, M_{Avg}, over all data which is found to be 5. This value is used as a reference line in Figure 2. Out of the 5 competing algorithms, FocusM, ABB and SetCover are deterministic, whereas LVF and QBB are non-deterministic due to their probabilistic nature. QBB spends half of the time running LVF and the other half running ABB. For LVF and QBB we show results for 5 different processing time in terms of total numbers of subsets evaluated (1000...5000). Each experiment was repeated 50 times. Notice that Focus and ABB are not shown in the graph as their average times fall outside the range of the 'processing time' in the x-axis of the graph, although minimal sized subsets are guaranteed in each case. For data having large differences between N and M values such as Lung Cancer, Promoters, Soybean, Splice data ABB takes very long time (a number of hours) to terminate. For data having large N values and not very small M values such as Splice data ($N = 60$, $M = 9$) FocusM takes many hours to terminate. The comparison in Figure 2 shows that QBB is more efficient both in average time and number of selected features compared to LVF, FocusM, and ABB. The average size of the subsets produced by QBB is close

to M_{Avg} and it approaches to M_{Avg} with time. SetCover produces near optimal subsets in much less time. Between QBB and SetCover we would say QBB is more robust while SetCover, although very fast and accurate, may fail to deliver efficient subsets if there is dependency among the features.

The error probability is often used as a validation criterion. Among the different algorithms discussed in the paper we take QBB due to its robustness. We choose C4.5 decision tree and Back-propagation neural network as two classifiers for validation. For back-propagation each data was divided into a training set (two-third of the original size) and the rest one-third as testing. For C4.5, we use the default settings, apply it to data before and after feature selection, and obtain the results of 10-fold cross-validation. This is repeated 10 times for each data and the average error rate and tree size are reported in Table 2. That is, QBB has been run 10 times and C4.5 100 times. The experiment shows the improvement/no reduction for most data (9 out of 10) in C4.5's accuracy after feature selection.

Running Back-propagation involves the setting of some parameters, such as the network structure (number of layers, number of hidden units), learning rate, momentum, number of CYCLES (epochs), etc. In order to focus our attention on the effect of feature selection by QBB, we try to minimize the tuning of the parameters for each data. We fix the learning rate at 0.1, the momentum at 0.5, one hidden layer, the number of hidden units as half of the original input units for all data. The experiment is carried out in two steps: (1) a trial run to find a proper number of CYCLES for each data which is determined by a sustained trend of no decrease of error; and (2) two runs on data with and without feature selection respectively using the number of CYCLES found in step 1. Other parameters remain fixed for the two runs in step 2. The results are shown in Table 2 with an emphasis on the difference before and after feature selection. In most cases, error rates decrease (6 out of 10) or do not change (3 out of 10) after feature selection.

	C4.5				Back-Propagation			
	Tree Size		Error Rate				Error Rate	
Data	Bef	Aft	Bef	Aft	Cycles	#HU	Bef	Aft
LED-24	19.0	19.0	0.0	0.0	1000	12	0.06	0.0
Lung	19.0	10.9	50.0	41.8	1000	28	75.0	75.0
Lymphography	26.9	22.1	21.8	21.4	7000	9	25.0	25.0
Mushroom	36.3	34.2	0.0	0.0	5000	11	0.0	0.0
Par3+3	12.0	15.0	41.9	0.0	1000	6	22.2	0.0
Promoters	21.4	8.2	26.3	22.1	2000	29	46.8	25.0
Soybean	7.1	9.4	2.5	2.0	1000	18	10.0	0.0
Splice	173.0	187.0	5.9	14.0	6000	30	25.64	42.33
Vote	14.5	14.2	5.3	5.3	4000	8	6.7	4.0
Zoo	17.8	17.7	7.8	6.6	4000	8	10.3	3.4

Table 2. Results of Hybrid Search. #HU is number of Hidden Units.

5 Concluding Remarks

The fact that the consistency criterion does not incorporate any search bias relating to a particular classifier enables it to be used with a variety of different learning algorithms. As shown in the experiments, for the two different types of classifiers, selected features improve the performance in terms of lower error rates in most cases. Features selected without search bias bring us efficiency in later stage as the evaluation of a feature subset becomes simpler than that of a full set. On the other hand, since a set of features is deemed consistent if *any* function maps from the values of the features to the class labels, any algorithm optimizing this criterion may choose a small set of features that has a complicated function, while overlooking larger sets of features admitting simple rules. Although intuitively this should be relatively rare, it can happen in practice, as apparently this was the case for the Splice data where both C4.5 and Back-propagation's performance deteriorate after feature selection.

The inconsistency measure has received a comprehensive examination that reveals its many merits for feature selection. The outstanding one is its monotonicity. It is also fast to compute, can detect redundant as well as irrelevant features. It has been used with a variety of search strategies in feature selection and no modification is required. The salient contribution of this work is that a guideline is suggested as to when to use what after detailed evaluation of different search algorithms. We believe the guideline will be very helpful to practitioners in need to reduce dimensionality of huge data, and to researchers who want to further the work of feature selection.

References

1. H. Almuallim and T. G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2):279–305, November 1994.
2. M. Ben-Bassat. Pattern recognition and reduction of dimensionality. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics*, pages 773–791. North Holland, 1982.
3. A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.
4. A Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Readings in Machine Learning*, pages 201–204, 1990.
5. G. Brassard and P. Bratley. *Fundamentals of Algorithms*. Prentice Hall, New Jersy, 1996.
6. M. Dash. Feature selection via set cover. In *Proceedings of* IEEE *Knowledge and Data Engineering Exchange Eorkshop*, pages 165–171, Newport, California, November 1997. IEEE Computer Society.
7. M. Dash and H. Liu. Feature selection methods for classification. *Intelligent Data Analysis: An Interbational Journal*, 1(3), 1997.
8. P. A. Devijver and J. Kittler. *Pattern Recognition : A Statistical Approach*. Prentice Hall, 1982.

9. G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, 1994.
10. D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
11. K. Kira and L. A. Rendell. The feature selection problem : Traditional methods and a new algorithm. In *Proceedings of Ninth National Conference on AI*, pages 129–134, 1992.
12. R. Kohavi. *Wrappers for performance enhancement and oblivious decision graphs.* PhD thesis, Department of Computer Science, Stanford University, CA, 1995.
13. H. Liu, H. Motoda, and M. Dash. A monotonic measure for optimal feature selection. In *Proceedings of European Conference on Machine Learning*, pages 101–106, 1998.
14. H. Liu and R. Setiono. Feature selection and classification - a probabilistic wrapper approach. In *Proceedings of Ninth International Conference on Industrial and Engineering Applications of AI and ES*, 1996.
15. C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1996. FTP from ics.uci.edu in the directory pub/machine-learning-databases.
16. P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature selection. *IEEE Transactions on Computers*, C-26(9):917–922, September 1977.
17. J. R. Quinlan. *C4.5 : Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.
18. T. W. Rauber. *Inductive Pattern Classification Methods - Features - Sensors.* PhD thesis, Department of Electrical Engineering, Universidale Nova de Lisboa, 1994.
19. W. Siedlecki and J Sklansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2:197–220, 1988.
20. S. Watanabe. *Pattern Recognition: Human and Mechanical.* Wiley Intersceince, 1985.
21. A. Zell and et al. Stuttgart Neural Network Simulator (SNNS), user manual, version 4.1. Technical report, 1995.

Feature Selection for Clustering

Manoranjan Dash and Huan Liu

School of Computing, National University of Singapore, Singapore.

Abstract. Clustering is an important data mining task. Data mining often concerns large and high-dimensional data but unfortunately most of the clustering algorithms in the literature are sensitive to largeness or high-dimensionality or both. Different features affect clusters differently, some are important for clusters while others may hinder the clustering task. An efficient way of handling it is by selecting a subset of important features. It helps in finding clusters efficiently, understanding the data better and reducing data size for efficient storage, collection and processing. The task of finding original important features for unsupervised data is largely untouched. Traditional feature selection algorithms work only for supervised data where class information is available. For unsupervised data, without class information, often principal components (PCs) are used, but PCs still require all features and they may be difficult to understand. Our approach: first features are ranked according to their importance on clustering and then a subset of important features are selected. For large data we use a scalable method using sampling. Empirical evaluation shows the effectiveness and scalability of our approach for benchmark and synthetic data sets.

1 Introduction

Clustering is an important data mining task that groups similar objects together [8,11,10,4,2]. Similarity between a pair of data points is due to different features. If similarity is distance-based then for a pair of data points in a cluster there exist at least a few features on which the points are close to each other. Most clustering methods assume all features to be equally important for clustering, or in other words they do not distinguish among different features. This is one of the reasons why most clustering algorithms may not perform well in the face of high-dimensional data. Another reason of the poor performance is the inherent sparsity of data in high-dimensional space. In reality different features have varying effects on clustering. An important feature helps in creating clusters while an unimportant feature may not help in creating clusters and, in contrary, it may affect the clustering algorithms adversely by blurring the clusters. Unimportant features are *noisy or irrelevant* and can be removed to reduce the data size for more efficient clustering. It also reduces the noise and helps in data storage, collection, and processing.

As clustering is done on unsupervised data without class information, traditional feature selection algorithms for classification [6] do not work. Little work

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 110–121, 2000.
© Springer-Verlag Berlin Heidelberg 2000

has been done on feature selection for unsupervised data. Dimensionality reduction or feature extraction methods (e.g., Principal Components Analysis, Karhunen-Loeve transformation, or Singular Value Decomposition) are commonly used [8]. They have drawbacks such as: (1) it is difficult to understand the data (and the found clusters) using the extracted features, and (2) the original features remain as they are required to determine the extracted features.

Some recent works on clustering try to handle high-dimensionality by selecting important features. In [2] and later in [5] it is observed that dense regions may be found in subspaces of high dimensional data. The algorithm called CLIQUE in [2] divides each dimension into a user given divisions. It starts with finding dense regions in 1-dimensional data and works upward to find k-dimensional dense regions using candidate generation algorithm Apriori [3]. This approach is different from the conventional clustering that partitions the whole data. In [1] a new concept is presented called "projected clustering" to discover interesting patterns in subspaces of high-dimensional data. It finds the clusters first and then selects a subset of features for each cluster. It searches for the subset of features by putting a restriction on the minimum and the maximum number of features.

We address the problem of selecting a subset of important features for clustering for the *whole data and not just for clusters* unlike in [1,2]. This helps in knowing the important features before doing clustering and the clustering task becomes more efficient and focused as only the important features can be used. Finding the important original features for the whole data helps in understanding the data better unlike principal components. Data storage, collection and processing tasks become more efficient and noise is reduced as the data is pruned.

Our approach is a 2-step method: we first rank and then select a subset of important features. Ranking of features is done according to their importance on clustering. An entropy-based ranking measure is introduced. We then select a subset of features using a criterion function for clustering that is invariant with respect to different numbers of features. A novel scalable method based on random sampling is introduced for large data commonly found in data mining applications.

2 Importance of Features on Clustering

Notations used in the paper are as follows: X_i is i^{th} data point, X_{ik} is k^{th} feature value of i^{th} point, F_k is k^{th} feature where $i = 1...N$ and $k = 1...N$; D_{i_1,i_2} and S_{i_1,i_2} are distance and similarity between points X_{i_1} and X_{i_2}; χ_j is j^{th} cluster where $j = 1...c$. We start by showing visually the effects of features on clustering. In Figure 1(a,b,c) we show a synthetic data in (3,2,1)-d feature spaces respectively. There are 75 points with 3 clusters in $F1$-$F2$ dimensions, with each cluster having 25 points. Values in $F1$ and $F2$ features follow Gaussian distribution within each of the 3 clusters while values in feature $F3$ are uniformly random. When we take 3 features the clusters are unclear and unnecessarily

complex (see Figure 1(a)), whereas no clusters can be found when we visualize using only 1 feature $F1$ (Figure 1(c)). Figure 1(b) with $F1$-$F2$ features shows 3 well-formed clusters. Selecting features $F1$ and $F2$ reduces the dimensionality of the data while forming well separated clusters.

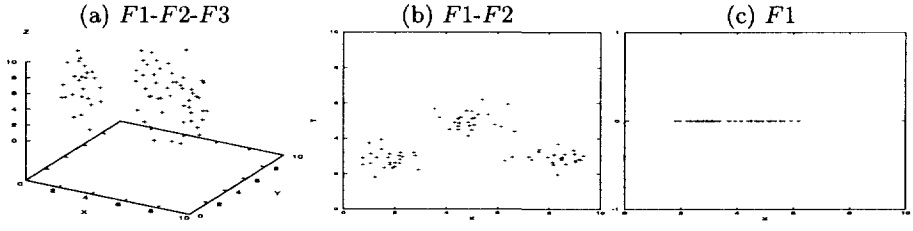


Fig. 1. Effect of features on clustering.

In a single dimensional data set clusters can be formed if the single feature takes values in separate ranges. In a multi-dimensional data set clusters can be formed from combination of feature values although the single features by themselves alone may take uniform values. We have noted down 2 distinct scenarios in the following.

Scenario 1: *A single feature is important by itself only*: Consider Figure 2(a) where there are 2 features. Feature $F2$ is uniformly distributed while $F1$ takes values in 2 separate ranges. It can be clearly seen that $F1$ is more important for creating clusters than $F2$.

Scenario 2: *Two features are necessarily important and any individual feature is useless in defining clusters*: Consider Figure 2(b) where there are 2 features. Both $F1$ and $F2$ are uniformly distributed. It can be clearly seen that both $F1$ and $F2$ are necessary for clustering and any one alone is useless.

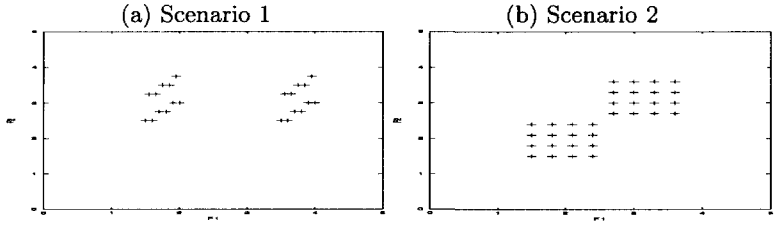


Fig. 2. Effects of features on clusters: scenario 1 shows effect of individual feature while scenario 2 shows the combined effect of 2 features.

3 Entropy-Based Feature Ranking

Consider each feature F_i as a random variable while f_i as its value. From entropy theory we know that, entropy is:
$E(F_1, ..., F_M) = -\sum_{f_1} ... \sum_{f_M} p(f_1, ..., f_M) \log p(f_1, ..., f_M)$ where $p(f_1, ..., f_M)$ is the probability or density at the point $(f_1, ..., f_M)$. If the probability is uniformly distributed we are most uncertain about the outcome, and entropy is maximum. This will happen when the data points are uniformly distributed in the feature space. On the other hand, when the data has well-formed clusters the uncertainty is low and so also the entropy. As we do not have a priori information about clusters, calculation of $p(f_1, ..., f_M)$ is not direct. But we can use the following way to calculate entropy without any cluster information.
Entropy Measure: Usually in a real-world data there may be a few not very well-formed clusters and some noise (points not belonging to any cluster properly). Two points belonging to the same cluster or 2 different clusters will contribute to the total entropy less than if they were uniformly sepatated. Similarity S_{i_1,i_2} between 2 instances X_{i_1} and X_{i_2} is high if the 2 instances are very close and S_{i_1,i_2} is low if the 2 are far away. Entropy E_{i_1,i_2} will be low if S_{i_1,i_2} is either low or high and E_{i_1,i_2} will be high otherwise. The following mathematical formulation is based on this idea.

Our similarity measure is applicable to both numeric and nominal data. Similarity is based on distance, i.e., for numeric data we use *Euclidean* distance while for nominal data we use *Hamming* distance. Mathematically similarity for numeric data is given as: $S_{i_1,i_2} = e^{-\alpha \times D_{i_1,i_2}}$ where α is a parameter. In a multi-dimensional space, distance D_{i_1,i_2} for numeric data is defined as: $D_{i_1,i_2} = [\sum_{k=1}^{M} (\frac{x_{i_1k} - x_{i_2k}}{max_k - min_k})^2]^{1/2}$. The interval in the k^{th} dimension is normalized by dividing it by the maximum interval $(max_k - min_k)$ before calculating the distance. If we plot similarity against distance, the curve will have a bigger curvature for a larger α. The insight is we assign a very high similarity for points 'very close' together but assign a low similarity for points 'not close' or 'far away'.

Similarity for *nominal* features is measured using the Hamming distance. The similarity between two data points is given as: $S_{i_1,i_2} = \frac{\sum_{k=1}^{M} |x_{i_1k} = x_{i_2k}|}{M}$ where $|x_{i_1k} = x_{i_2k}|$ is 1 if x_{i_1k} equals x_{i_2k} and 0 otherwise. For data with both numeric and nominal features, we can discretize numeric values first before applying our measure. For two points X_{i_1} and X_{i_2}, entropy is: $E = -S_{i_1,i_2} \log S_{i_1,i_2} - (1 - S_{i_1,i_2}) \log (1 - S_{i_1,i_2})$ which assumes the maximum value of 1.0 for $S_{i_1,i_2} = 0.5$, and the minimum value of 0.0 for $S_{i_1,i_2} = 0.0$ and $S_{i_1,i_2} = 1.0$. For a data set of N data points entropy is given as: $E = -\sum_{i_1=1}^{N} \sum_{i_2=1}^{N} (S_{i_1,i_2} \times \log S_{i_1,i_2} + (1 - S_{i_1,i_2}) \times \log(1 - S_{i_1,i_2}))$ where S_{i_1,i_2} takes values in [0.0-1.0]. In this work, α is calculated automatically by assigning 0.5 in Equation: $\overline{S} = e^{-\alpha \times \overline{D}}$ at which entropy is maximum; so we get: $\alpha = \frac{-\ln 0.5}{\overline{D}}$ where $\overline{D}$ is the average distance among the data points.
Algorithm to Rank Features: If the removal of feature F_1 causes more disorder than the removal of feature F_2 then $E_{-F_1} > E_{-F_2}$ where E_{-F_1} and E_{-F_2}

are entropy after removing F_1 and F_2 respectively. In scenario 1 (Figure 2(a)) $E_{-F_1} > E_{-F_2}$. For scenario 2 (Figure 2(b)) we added one more feature F_3 which takes uniformaly random values and does not take part in forming the 2 clusters. As expected we got $E_{-F_1} > E_{-F_3}$ and $E_{-F_2} > E_{-F_3}$. *Secenario 2 suggests that our entropy measure works for dependent features also.*

For ranking of features we can use E in the following way: Each feature is removed in turn and E is calculated. If the removal of a feature results in minimum E the feature is the least important; and *vice versa.* In the algorithm CalcEnt(F_k) calculates E of the data after discarding feature F_k.

Algorithm (RANK):

$P = E$ values for M features
For $k = 1$ to M
  P_k = CalcEnt(F_k)
OutputRank(P)

Scalable Feature Ranking: Data mining generally concerns data with large number of data points. For a large number of data points our ranking measure may not be practical *as it is* (the complexity of RANK is $O(MN^2)$ if we take the similarity measure between 2 points as unit). There are different approaches available in the literature for handling large data sets for a given algorithm. Our scalable method is based on *random sampling.* We observed that a reasonably small random sample retains the original cluster information in most cases. This phenomenon was also observed in [9] in their work on initialization of partitional clustering algorithms. Notice that for entropy measure to work well the cluster structure needs to be retained and it is largely independent of the number of data points. So, random sampling is a good choice for scalability. The algorithm is simple. Initially all features are ranked 0. Random samples are generated and RANK is run over each sample to produce the rankings of features. The feature rankings are added correspondingly. At the end of all random samples p (we suggest the use of at least 35 samples as 35 is often considered the minimum number of samples for large sample procedures [7]) we obtain the final rankings of the features.

Algorithm for Scalable Ranking, SRANK

for all features F_k's *O*verall *R*ank, $OR_k = 0$
for $l = 1$ to p
  take a sample L_l
  run RANK to find rankings R_l
  for $k = 1$ to M
    $OR_k = OR_k + R_{l_k}$
output overall rankings OR

Selecting a Subset of Important Features:

A problem is how many features we should choose from a ranked list. A natural expectation is that entropy would fall initially with removal of unimportant features, but would stop falling at some point. This is not the case as the entropy

is not invariant with respect to different numbers of features. Hence we are left with finding the different alternatives of selecting a subset of features: (1) If one knows the number of important features required, just pick them starting with the most important one, or (2) we can choose a clustering algorithm and choose the subset that maximizes the clustering quality. The first option is not practical without any a priori knowledge. The second option is a wrapper method. Wrapper method is a feature selection method that wraps around clustering algorithm which is a standard way for supervised feature selection with a classification algorithm [12]. The difference is that *in our case features are already ranked according to their importance and so the task of searching through the feature subset space of* 2^M *is avoided.* The idea is to run a clustering algorithm on the selected features and choose the subset that produces best cluster quality.

We choose k-means clustering algorithm which is very popular and simple to implement. It is iterative, provides results fast (converges fast to local maxima), has a time complexity of $\#Iter * |Data| * c$ where $\#Iter$ is is the number of iterations, $|Data|$ is the size of the data $(N * M)$, and c is the number of clusters. Once the clustering is done we need to measure the cluster quality. There are numerous criterion functions for clustering in literature to measure cluster quality. We select scattering criterion which is invariant under nonsingular transformation of data. **Scattering Criteria**: These criteria consider the scatter matrices used in multiple discriminant analysis. Scatter matrix for j^{th} cluster: $P_j = \sum_{X_i \in X_j} (X_i - m_j)(X_i - m_j)^t$ Within-cluster scatter matrix: $P_W = \sum_{j=1}^{c} P_j$ Between-cluster scatter matrix: $P_B = \sum_{j=1}^{c} (m_j - m)(m_j - m)^t$ where m is the total mean vector and m_j is the mean vector for j^{th} cluster and $(X_i - m_j)^t$ is the matrix transpose of the column vector $(X_i - m_j)$. Among different scattering criteria we briefly describe here an 'Invariant Criterion'. One invariant criterion is: $tr({P_W}^{-1} P_B)$ where tr is *trace* of a matrix which is the sum of its diagonal elements. It is invariant under nonsingular linear transformations of the data. It measures the ratio of between-cluster to within-cluster scatter. The higher the $tr({P_W}^{-1} P_B)$, the higher the ratio of between-cluster scatter to within-cluster one and hence, and hence, the higher the cluster quality. We use $tr({P_W}^{-1} P_B)$ to compare the cluster quality for different subsets of important features. The algorithm for selecting a subset of features is as follows:

Algorithm SELECT:

run RANK to get rankings $R_k, k = 1...M$
for k=1 to M
run K-means to find clusters using subset $(R_1, ..., R_k)$
calculate $tr(P_W^{-1} P_B)$
if stopping criterion satisfy break

In case of large data run SRANK instead of RANK. $Tr(P_W^{-1} P_B)$ will increase with the addition of features if the ratio of between-cluster to within-cluster increases, otherwise it decreases or remains relatively unchanged. As is found by the experiments, $tr(P_W^{-1} P_B)$ increases initially and once all important features are added, it either goes down or remains relatively unchanged for any addition of unimportant features. The point at which $tr(P_W^{-1} P_B)$ goes down or

remains unchanged is not difficult to detect visually, hence the stopping criterion is manually decided.

4 Experiments

We empirically tested our feature selection method on different scenarios that one may find in various data mining applications. First, tests are conducted on benchmark and synthetic data sets to check the correctness of our claim that our feature selection method can select correct features as we know well about these data sets. Tests are then conducted on a large high-dimensional data to test the performance of SRANK. We used a MATLAB random function to generate synthetic data. For synthetic data sets a few features are chosen as important and these features follow Gaussian distribution. Each cluster is of equal size if not mentioned otherwise. Clusters are usually overlapping. Unimportant features are added which take uniformly random values. Each data has 5% noisy data points.

Benchmark and Synthetic Data: Three synthetic data sets are generated with different numbers of clusters and features. Benchmark data sets (both numeric and nominal) are selected from UCI machine-learning repository [13]. See Table 1 for the details about data sets. We have chosen those data sets from the repository for which prior information is available regarding importance of features. Although for these benchmark data sets class information is available, in our experiments we have removed the class labels. Parity3+3 has 3 relevant, 3 redundant, and 6 irrelevant features.

The results for ranking are shown in Table 1. Our method is able to rank the important features in the top ranks for all data. For CorrAL our method ranks feature $F6$ higher. $F6$ is correlated to the class label 75% of the data points. This shows our ranking measure favors features that are correlated to the class. Although for CorrAL this is not desired but for real-world data this may be acceptable. For Parity3+3 ranking was correct although the redundant features could not be detected. This can be removed if after selecting a subset of features we check for redundancy between the features in pair. For a small subset of selected features this may not be extermely prohibitive.

The results for selecting a subset of features are shown for the data sets with a known number of clusters and numeric data. We use k-means and $tr({P_W}^{-1}P_B)$ to evaluate the subsets of important features. Initialization of k-means is done by randomly choosing points from the data. Once a set of points are chosen for the whole data the same set is used for different subsets of features. The results are summarized in Figure 3. The X-axis of the plots is for number of most important features and Y-axis is for $tr(P_W^{-1}P_B)$ value for the corresponding subset of most important features. For Iris data set *trace* value was the maximum for the two most important features. For D3C, D4C and D6C data *trace* value increases with addition of important features in a fast rate but slows down to almost a halt after all the important features are added. For a practical application it will not

Data Set	M	#Clusters/Classes	Important Features	Ranking (Descending Order)
Iris	4	3	3,4	**{3,4}**,1,2
ChemicalPlant	5	∞	1,2,3	**{3,1,2}**,4,5
Non-linear	4	∞	1,2	**{2,1}**,3,4
Parity3+3	12	2	{1,7},{2,8},{3,9}	**{9,3,8,2,7,1}**,4,10,...
CorrAL	6	2	1,2,3,4	**{3,6,1,2,4}**,5
Monk3	6	2	2,4,5	**{5,2,4}**,1,6,3
D3C	4	3	1,2	**{2,1}**,4,3
D4C	15	4	1-5	**{1,3,5,2,4}**,13,9,11,...
D6C	22	6	1-7	**{3,6,5,4,2,1,7}**,10,9,...

Table 1. Ranking of features: ∞ – Class is continuous, bold font is used to show the correctness of the ranking.

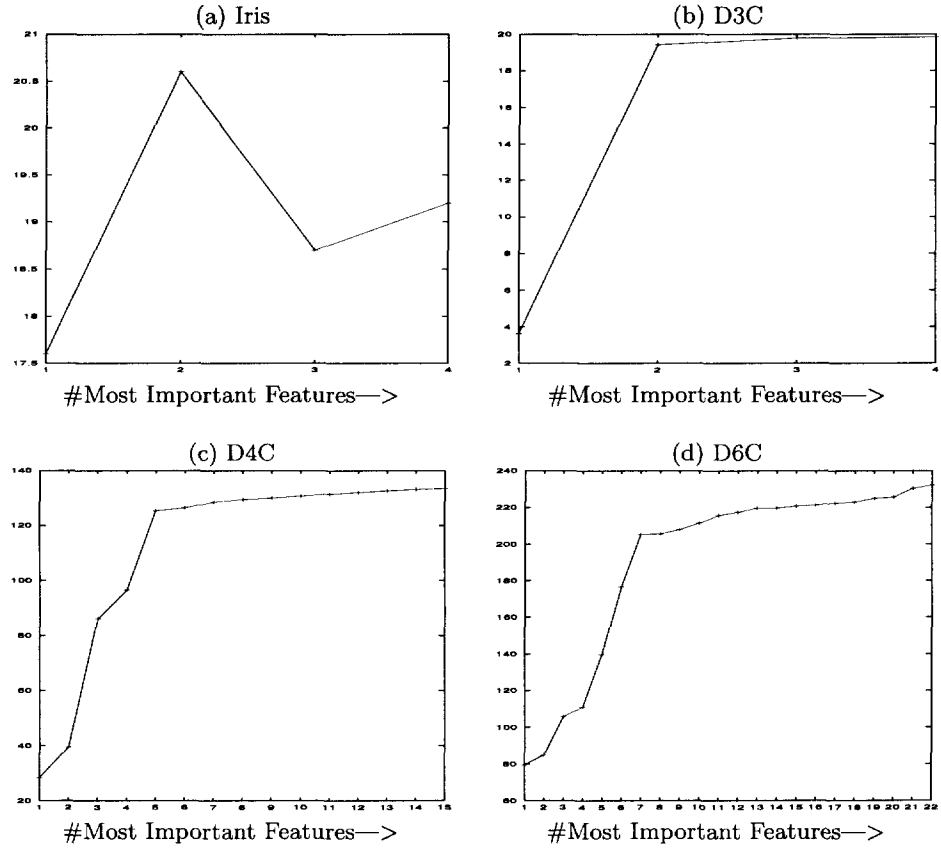


Fig. 3. $tr(P_W^{-1}P_B)$ of Iris and Synthetic data.

be difficult to notice these trends, and hence selecting a subset of features can be an easy task.

Large and High-Dimensional Data: We show the results of our feature selection on a synthetic large and high-dimensional data. The data has 100 features (first 20 features are important and the next 80 features unimportant), 5 clusters, each cluster created by Gaussian distribution, unimportant features take uniformly random values. Each cluster has 20,000 points and the data has 5000 (approximately 5%) noisy data points. Sample sizes chosen are 0.25%, 0.50% and 1.0%.

Results: For space constraint we have shown results of SRANK and SELECT for 5 samples. SRANK results are shown in Table 2. The last row in Table 2 is the over all rankings after 5 runs. In all the runs the 20 important features are ranked at the top, and hence, they are ranked at the top in over all ranking as well. SELECT results are shown in Figure 4 and Table 3. We have shown average results for 5 sample runs for 0.25%. In Table 3 impurity is "number of misclassifications not including the noise"[1]. Notice that impurity is low or zero when only important features are used and it grows with addition of unimportant features. It further confirms our suspicion that k-means (and probably other clustering algorithms) get confused with useless features and removing them can contribute to the cluster quality. This result shows satisfactory scalability of SRANK and SELECT.

5 Conclusion

We tested RANK over a real-world textual finance data. As many as 423 phrases or words are used as features for each textual financial data taken on a daily basis from reliable and standard sources such as Wall Stree Journal. The feature values are the frequencies of the corresponding phrases in that day's reports. After running RANK over this high-dimensional data we showed the results to a domain expert. He was satisfied regarding the top rankings given to important phrases such as: blue chip, property lost, banking retreat, etc. Efforts are on to use this ranking for prediction purposes. Another application yet untested is Reuters text categorization data which has hundereds of thousands of words as features. It may be useful to pick up a few hundred words for further classification. We studied a number of related issues such as high-dimensional data, noisy data, large data, redundant/correlated features, and hill-climbing *vs.* exhaustive. Handling high dimensional data is a prominent desirable characteristic of our method. Experiments show that in the face of high-dimensional data k-means algorithm perform poorly, but removal of unimportant features significantly improved its performance. Our method is able to handle noisy data. To handle very large data sets we used random samples. Our ranking measure works well consistently for

[1] As we have generated the data, the data points that group together in a cluster are known and it enables us to find impurity after clustering. This may not be the case for real-world data, and hence $tr(P_W{}^{-1}P_B)$ is more useful practically.

#Run	Sample Size		
	0.25%	0.50%	1.0%
1	{15,5,20,14,9,12,2, 7,18,11,17,1,19,10,3, 6,16,8,4,13},42,57,...	{20,5,19,3,14,17,9, 2,11,10,13,1,16,7,4, 8,15,6,18,12},43,81,...	{15,19,3,16,13,10,9,5, 11,17,12,4,14,20,2,1, 8,18,6,7},71,23,...
2	{5,6,14,20,7,10,12, 17,16,18,15,13,8,9,19, 2,11,1,4,3},63,25,...	{19,14,16,13,3,6,15, 18,17,2,11,8,1,4,7, 9,5,12,10,20},55,23,...	{12,13,7,6,4,1,19,3, 9,20,10,11,15,18,8,2, 14,17,16,5},42,29,...
3	{14,6,17,13,12,9,20,15, 10,5,2,19,1,16,8,7, 11,3,18,4},29,92,...	{19,10,15,2,18,3,8, 13,16,7,17,14,12,5,11, 20,9,4,1,6},68,39,...	{9,4,5,2,16,14,1,3, 12,19,20,6,17,15,18,10, 13,7,8,11},71,92,...
4	{11,12,17,1,4,9,8,3, 5,18,16,2,6,19,14,13, 7,20,15,10},32,77,...	{8,1,3,19,15,18,12,7,11, 2,4,20,10,13,5,14,17, 6,9,16},44,83,...	{10,19,12,6,1,14,8, 7,20,17,18,13,16,15,2, 4,11,5,3,9},26,70,...
5	{13,19,9,16,20,18,10, 6,8,4,12,5,15,14,17, 2,1,11,3,7},42,37,...	{15,16,13,2,10,8,19,11,14, 4,3,6,1,9,17,12,7, 20,18,5},37,24,...	{4,15,14,13,3,9,10,19, 1,7,8,12,5,18,2,16, 17,6,20,11},74,53,...
OverAll Ranking	{12,5,9,14,20,17, 6,13,15,11,18,10,16,19, 2,8,1,7,3,4},23,98...	{19,3,15,2,13,8,14, 16,10,11,18,17,1,7,12, 20,4,5,9,6},45,87,...	{19,4,12,13,10,1,3, 9,15,14,16,6,7,5,20, 2,17,8,18,11},62,54,...

Table 2. Ranking for 5 random samples of 3 different sizes

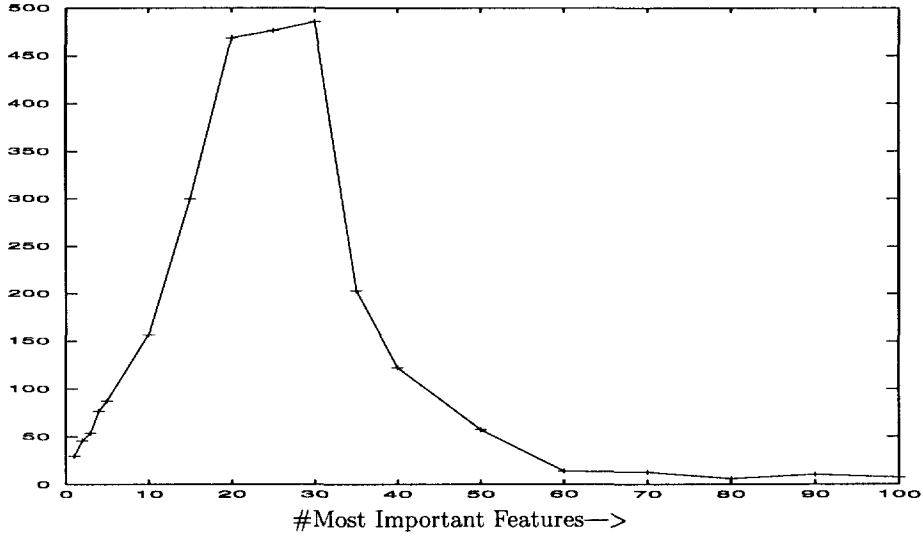


Fig. 4. Average $tr(P_W^{-1}P_B)$ of 5 samples of size 0.25% of a Large and High-Dimensional Data.

#MostImpFea	$tr(P_W^{-1}P_B)$	Impurity (%)	#MostImpFea	$tr(P_W^{-1}P_B)$	Impurity (%)
1	29.56	10.4	35	203.18	10.2
5	87.13	8.2	40	122.5	17.4
10	156.4	0.0	50	57.4	36.8
15	299.67	0.0	60	13.71	62.2
20	468.81	0.0	70	5.87	56.0
25	476.36	0.0	80	10.37	66.4
30	485.57	2.0	100	7.13	73.0

Table 3. Average $tr(P_W^{-1}P_B)$ and Impurity of 5 samples of 0.25% of a large and high dimensional Data

the different runs with different sizes of random samples. Our method only requires the cluster structure be retained which a reasonably small random sample is expected to maintain. We studied the issue of redundant/correlated features. We did an experimental study of comparing our hill-climbing feature selection method *vis-a-vis* exhaustive method. Hill-climbing method performed reliably while consuming much less time.

Testing our feature selection method for clustering algorithms other than k-means is an ongoing work. But as shown by the experiments over data sets with known important features, it can be expected that our algorithm would perform equally well for other clustering algorithms. Another area to explore is subspace clustering (CLIQUE [2]) which is the task of finding dense regions in subspaces of features instead of whole space. It can help find interesting data hidden in subspaces of features where clusters may not be defined by all features. A problem encountered in CLIQUE concerns scalability with respect to number of features. Their experiments exhibited a quadratic behavior in the number of features. It may be interesting to check the effectiveness of our approach in reducing the dimensionality thereby making the search for subspace clusters more efficient.

References

1. C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 61–72, 1999.
2. R Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1998.
3. R Agrawal and R. Srikant. Fast algorithm for mining association rules. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.
4. P. S. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of the 4th International Conference on Knowledge Discovery & Data Mining (KDD'98)*, pages 9–15, 1998.

5. C. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of Internationl Conference on Knowledge Discovery and Data Mining (KDD'99)*, 1999.
6. M. Dash and H. Liu. Feature selection for classification. *International Journal of Intelligent Data Analysis, http://www.elsevier.com/locate/ida*, 1(3), 1997.
7. J. L. Devore. *Probability and Statistics for Engineering and Sciences.* Duxbury Press, 4th edition, 1995.
8. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*, chapter Unsupervised Learning and Clustering. John Wiley & Sons, 1973.
9. U. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinment clustering algorithms. In *Proceedings of the 4th International Conference on Knowledge Discovery & Data Mining (KDD'98)*, pages 194–198, 1998.
10. V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS - clustering categorical data using summaries. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD'99)*, 1999.
11. A. K. Jain and R. C. Dubes. *Algorithm for Clustering Data*, chapter Clustering Methods and Algorithms. Prentice-Hall Advanced Reference Series, 1988.
12. R. Kohavi. *Wrappers for performance enhancement and oblivious decision graphs.* PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 1995.
13. C. J. Merz and P. M. Murphy. UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1996.

A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases

Eamonn J. Keogh and Michael J. Pazzani

Department of Information and Computer Science
University of California, Irvine, California 92697 USA
{eamonn,pazzani}@ics.uci.edu

Abstract. We address the problem of similarity search in large time series databases. We introduce a novel-dimensionality reduction technique that supports an indexing algorithm that is more than an order of magnitude faster than the previous best known method. In addition to being much faster our approach has numerous other advantages. It is simple to understand and implement, allows more flexible distance measures including weighted Euclidean queries and the index can be built in linear time. We call our approach PCA-indexing (Piecewise Constant Approximation) and experimentally validate it on space telemetry, financial, astronomical, medical and synthetic data.

1 Introduction

Recently there has been much interest in the problem of similarity search in time series databases. This is hardly surprising given that time series account for much of the data stored in business, medical and scientific databases. Similarity search is useful in its own right as a tool for exploring time series databases, and it is also an important subroutine in many KDD applications such as clustering [6], classification [14] and mining of association rules [5].

Time series databases are often extremely large. Given the magnitude of many time series databases, much research has been devoted to speeding up the search process [23,1,15,19,4,11]. The most promising methods are techniques that perform dimensionality reduction on the data, then use spatial access methods to index the data in the transformed space. The technique introduced in [1] and extended in [8, 21,23]. The original work by Agrawal et al. utilizes the Discrete Fourier Transform (DFT) to perform the dimensionality reduction, but other techniques have been suggested, most notably the wavelet transform [4].

In this paper we introduce a novel transform to achieve dimensionality reduction. The method is motivated by the simple observation that for most time series datasets we can approximate the data by segmenting the sequences into equi-length sections and recording the mean value of these sections. These mean values can be indexed efficiently in a lower dimensionality space. We compare our method to DFT, the only

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 122-133, 2000.
© Springer-Verlag Berlin Heidelberg 2000

obvious competitor, and demonstrate a one to two order of magnitude speedup on four natural and two synthetic datasets.

In addition to being much faster. We demonstrate that our approach has numerous other advantages over DFT. It is simple to understand and implement, allows more flexible queries including the weighted Euclidean distance measure, and the index can be built in linear time. In addition our method also allows queries which are shorter than length for which the index was built. This very desirable feature is impossible in DFT and wavelet transforms due to translation invariance [20].

The rest of the paper is organized as follows. In Section 2, we state the similarity search problem more formally and survey related work. In Section 3, we introduce our method. Section 4 contains extensive empirical evaluation of our technique. In Section 5, we demonstrate how our technique allows more flexible distance measures. Section 6 offers concluding remarks and directions for future work.

2 Background and Related Work

Given two sequences $X = x_1 \ldots x_n$ and $Y = y_1 \ldots y_m$ with $n = m$, their Euclidean distance is defined as:

$$D(X,Y) \equiv \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

There are essentially two ways the data might be organized [8]:

- *Whole Matching.* Here it assumed that all sequences to be compared are the same length.
- *Subsequence Matching.* Here we have a query sequence X, and a longer sequence Y. The task is to find the subsequence in Y, beginning at Y_i, which best matches X, and report its offset within Y.

Whole matching requires comparing the query sequence to each candidate sequence by evaluating the distance function and keeping track of the sequence with the lowest distance. Subsequence matching requires that the query X be placed at every possible offset within the longer sequence Y. Note it is possible to convert subsequence matching to whole matching by sliding a "window" of length n across Y, and making copies of the m-n windows. Figure 1 illustrates the idea. Although this causes storage redundancy it simplifies the notation and algorithms so we will adopt this policy for the rest of this paper.

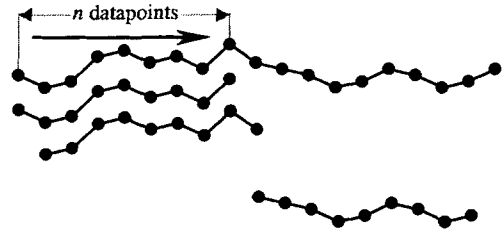


Figure 1: The subsequence matching problem can be converted into the whole matching problem by sliding a "window" of length n across the long sequence and making copies of the data falling within the windows

There are several kinds of queries that could be made of a database, such as range queries, all-pairs and nearest neighbor. For simplicity, we will concentrate just on nearest neighbor. The other kinds of queries can always be built using nearest neighbor, and the extensions are trivial.

Given a query X and a database consisting of K time series Y_i $(1 \leq i \leq K)$, we want to find the time series Y_i such that $D(Y_i,X)$ is minimized. The brute force approach, sequential scanning, requires comparing every time series Y to X. Clearly this approach is unrealistic for large datasets.

Any indexing scheme that does not examine the entire dataset could potentially suffer from two problems, false alarms and false dismissals. False alarms occur when objects that appear to be close in the index are actually distant. Because false alarms can be removed in a post-processing stage (by confirming distance estimates on the original data), they can be tolerated so long as they are relatively infrequent. In contrast, false dismissals, when qualifying objects are missed because they appear distant in index space, are usually unacceptable. In this work we will focus on admissible searching, indexing techniques that guarantee no false dismissals.

2.1 Related Work

A time series X can be considered as a point in n-dimensional space. This immediately suggests that time series could be indexed by Spatial Access Methods (SAMs) such as the R-tree and its many variants [9,3]. However SAMs begin to degrade rapidly at dimensionalities greater than 8-10 [12], and realistic queries typically contain 20 to 1,000 datapoints. In order to utilize SAMs it is necessary to first perform dimensionality reduction. Several dimensionality reduction schemes have been proposed. The first of these *F-index*, was introduced in [1] and extended in [8,23,21]. Because this is the current state-of-the-art for time series indexing we will consider it in some detail.

An important result in [8] is that the authors proved that in order to guarantee no false dismissals, the distance in the index space must satisfy the following condition

$$D_{\text{true}}(A,B) \geq D_{\text{index space}}(A,B) \tag{2}$$

Given this fact, and the ready availability of off-the-shelf SAMs, a generic technique for building an admissible index suggests itself. Given the true distance metric (in this case Euclidean) defined on n datapoints, it is sufficient to do the following:

- Produce a dimensionality reduction technique that reduces the dimensionality of the data from n to N, where N can be efficiently handled by your favorite SAM.
- Produce a distance measure defined on the N dimensional representation of the data, and prove that it obeys $D_{\text{true}}(A,B) \geq D_{\text{index space}}(A,B)$.

In [8] the dimensionality reduction technique chosen was the Discrete Fourier Transform (DFT). Each of the time series are transformed by the DFT. The Fourier representation is truncated, that is, only the first k coefficients are retained $(1 \leq k < n)$, and the rest discarded. The k coefficients can then be mapped into $2k$ space ($2k$ because each coefficient has a real and imaginary component) and indexed by an R* tree.

An important property of the Fourier Transform is Parseval's Theorem, which states that the energy in Euclidean space is conserved in Fourier space [18]. Because of the truncation of positive terms the distance in the transformed space is guaranteed to underestimate the true distance. This property is exploited by mapping the query into the same $2k$ space and examining the nearest neighbors. The theorem guarantees underestimation of distance, so it is possible that some apparently close neighbors are actually poor matches. These false alarms can be detected by examining the corresponding original time series in a post processing stage.
Many other schemes have been proposed for similarity search in time series databases. As they focus on speeding up search by sacrificing the guarantee of no false dismissals [11, 15, 19], and/or allowing more flexible distances measures [2,11, 15, 14, 13, 23, 16, 21] we will not discuss them further.

3 Our Approach

As noted by Faloutsos et al. [8], there are several highly desirable properties for any indexing scheme:

- It should be much faster than sequential scanning.
- The method should require little space overhead.
- The method should be able to handle queries of various lengths.
- The method should be allow insertions and deletions without requiring the index to be rebuilt.
- It should be correct, i.e. there should be no false dismissals.

We will now introduce the *PCA* indexing scheme and demonstrate that it has all the above properties.

3.1 Dimensionality Reduction

We denote a time series query as $X = x_1,\ldots,x_n$, and the set of time series which constitute the database as $Y = \{Y_1,\ldots Y_K\}$. Without loss of generality, we assume each sequence in Y is n units long. Let N be the dimensionality of the transformed space we wish to index ($1 \le N \le n$). For convenience, we assume that N is a factor of n. This is not a requirement of our approach, however it does simplify notation.
A time series X of length n is represented in N space by a vector $\overline{X} = \overline{x}_1,\mathrm{K}\ ,\overline{x}_N$. The i^{th} element of $\overline{X}$ is calculated by the following equation:

$$\overline{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j \tag{3}$$

Simply stated, to reduce the data from n dimensions to N dimensions, the data is divided into N equi-sized "frames". The mean value of the data falling within a frame is calculated and a vector of these values becomes the data reduced representation. Figure 2 illustrates this notation. The complicated subscripting in Eq. 3 is just to insure that the original sequence is divided into the correct number and size of frames.

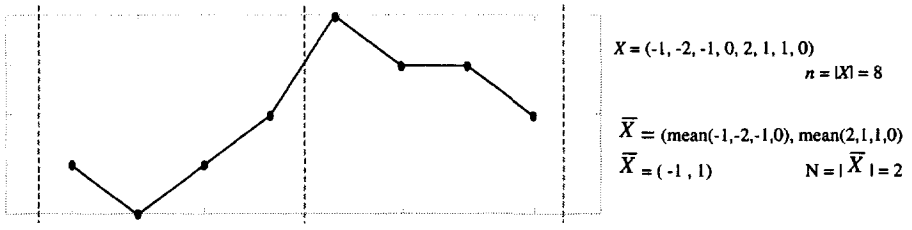


Figure 2: An illustration of the data reduction technique utilized in this paper. A time series consisting of eight (n) points is projected into two (N) dimensions. The time series is divided into two (N) frames and the mean of each frame is calculated. A vector of these means becomes the data reduced representation

Two special cases worth noting are when $N = n$ the transformed representation is identical to the original representation. When $N = 1$ the transformed representation is simply the mean of the original sequence. More generally the transformation produces a piecewise constant approximation of the original sequence.

3.2 Building the Index

Table 1 contains an outline of the indexing algorithm. We are deliberately non-committal about the particular indexing structure used. This is to reinforce the fact the dimensionality reduction technique proposed is independent of the indexing structure. All sequences in Y are transformed by Eq. 3 and indexed by the spatial access method of choice. The indexing tree represents the transformed sequences as points in N dimensional space. Each point contains a pointer to the corresponding original sequence on disk.

```
for i = 1 to K                          // For each sequence to be indexed
  Y_i  ← Y_i - mean(Y_i);               // Optional: remove the mean of Y_i
  Ȳ_i ← transformed(Y_i);               // As in eq. 3
  Insert Ȳ_i into the indexing structure with a pointer to Y_i
on disk;
end;
```

Table 1: An outline of the indexing building algorithm.

Note that each sequence has its mean subtracted before indexing. This has the effect of shifting the sequence in the y-axis such that its mean is zero, removing information about its offset. This step is optional. We include it because we want to compare our results directly to *F-index*, and *F-index* discards information about offset. For some applications this step is undesirable and can be omitted [13]. Note that the transformation for a single sequence takes $O(n)$ time, thus the entire index can be built in $O(Kn)$. This contrasts well to *F-index* which requires $O(KnLogn)$ time.

3.3 Searching the Index

As mentioned in Section 2, in order to guarantee no false dismissals we must produce a distance measure *DR*, defined in index space, which has the following property: $D(X,Y) \geq DR(\overline{X},\overline{Y})$. The following distance measure has this property:

$$DR(\overline{X},\overline{Y}) \equiv \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (\overline{x}_i - \overline{y}_i)^2} \tag{4}$$

The proof that $D(X,Y) \geq DR(\overline{X},\overline{Y})$ is straightforward but long. We omit it for brevity. Table 2 below contains an outline of the nearest neighbor search algorithm. Once a query X is obtained a transformed copy of it $\overline{X}$ is produced. The indexing structure is searched for the nearest neighbor of $\overline{X}$. The original sequence pointed to by this nearest neighbor is retrieved from disk and the true Euclidean distance is calculated. If the second closest neighbor in the index is further than this true Euclidean distance, we can abandon the search, because we are guaranteed its distance in the index is an underestimate of its true distance to the query. Failing that, the algorithm repeatedly retrieves the sequence pointed to by the next most promising item in the index and tests if its true distance is greater than the current best so far. As soon as that happens the search is abandoned.

```
best-so-far   ← infinity;
done          ← FALSE;
i             ← 1;
X̄             ← transformed(X);                    // Using eq.2
while i ≤ G AND NOT(done)
  Find X̄'s i^th nearest neighbor in the index;   // Using DR (eq.3)
  Retrieve sequence represented by the i^th nearest neighbor;
   if D(original-sequence_i, X) < best-so-far  //D is defined in eq.1
              best-so-far ← D(original-sequence_i, X);
   end;
   if best-so-far ≤ i^th+1 nearest neighbor in the index
              done   ← TRUE;
              Display('Sequence ',i,' is the nearest neighbor to Query');
              Display('At a distance of ',best-so-far );
   end;
     i ← i + 1;
  end;
```

Table 2: An outline of the indexing searching algorithm.

3.4 Handling Queries of Various Lenghts

In the previous section we showed how to handle queries of length n, the length for which the index structure was built. However, it is possible that a user might wish to query the index with a query which is longer or shorter that n. For example a user might normally be interested in monthly patterns in the stock market, but occasionally wish to search for weekly patterns. Naturally we wish to avoid building an index for

every possible length of query. In this section we will demonstrate how we can execute queries of different lengths on a single fixed-length index. For convenience we will denote queries longer than n as XL and queries shorter than n as XS, with $|XL| = n_{XL}$ and $|XS| = n_{XS}$.

3.4.1 Handling Short Queries

Queries shorter than n can be dealt with in two ways. If the SAM used supports dimension weighting (for example the hybrid tree [3]) one can simply weigh all the dimensions from $ceiling(\frac{N n_{XS}}{n})$ to N as zero. Alternatively, the distance calculation in Eq. 4 can have the upper bound of its summation modified to:

$$Nshort = \left\lceil \frac{N n_{XS}}{n} \right\rceil, \qquad \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{Nshort} (\bar{x}_i - \bar{y}_i)^2} \tag{5}$$

The modification does not affect the admissibility of the no false dismissal condition in eq. 2. Because the distance measure is the same as Eq. 4 which we proved, except we are summing over an extra 0 to $\frac{n}{N} - 1$ nonnegative terms on the larger side of the inequality. Apart from making either one of these changes, the nearest neighbor search algorithm given in table 2 is used unmodified. This ability of *PCA-index* to handle short queries is an attractive feature not shared by *F-index*, which must resort to sequential scanning in this case [8], as must indexing schemes based on wavelets [20].

3.4.2 Handling Longer Queries

Handling long queries is a little more difficult than the short query case. Our index only contains information about sequences of length n (projected into N dimensions) yet the query XL is of length n_{XL} with $n_{XL} > n$. However we can regard the index as containing information about the prefixes of potential matches to the longer sequence. In particular we note that the distance in index space between the prefix of the query and the prefix of any potential match is always less than or equal to the true Euclidean distance between the query and the corresponding original sequence. Given this fact we can use the nearest neighbor algorithm outlined in table 2 with just two minor modifications. In line four, the query is transformed into the representation used in the index, here we need to replace X with $XL[1{:}n]$. The remainder of the sequence, $XL[n{+}1{:}n_{XL}]$, is ignored during this operation.

In line seven, the original data sequence pointed most promising object in the index is retrieved. For long queries, the original data sequence retrieved and subsequently compared to XL must be of length n_{XL} not n.

4 Experimental Results

To demonstrate the generality of our method we tested it on five datasets with widely varying properties.

- **Random Walk:** The sequence is a random walk $x_t = x_{t-1} + z_t$ Where $z_t (t = 1,2,\ldots)$ are independent identically distributed (uniformly) random variables in the range (-500,500) [1]. (100,000 datapoints).

- **Astronomical:** A dataset that describes the rate of photon arrivals [17]. (28,904 datapoints).
- **Financial:** The US Daily 5-Year Treasury Constant Maturity Rate, 1972 - 1996 [15]. (8,749 datapoints).
- **Space Shuttle:** This dataset consists of ten time series that describe the orientation of the Space Shuttle during the first eight hours of mission STS-57 [14,15]. (100,000 datapoints).
- **Control Chart:** This dataset consists of the Cyclic pattern subset of the control chart data from the UCI KDD archive (kdd.ics.uci.edu). The data is essentially a sine wave with noise. (6,000 datapoints).

4.1 Building Queries

Choosing queries that actually appear in the indexed database will always produce optimistic results. On the other hand, some indexing schemes can do well if the query is greatly different from any sequence in the dataset. To perform realistic testing we need queries that do not have exact matches in the database but have similar properties of shape, structure, spectral signature, variance etc. To achieve this we do the following. We extract a sequence from the database then flip it either backwards or upside-down depending on the outcome of a fair coin toss. The flipped sequence then becomes our query.
For every combination of dataset, number of dimensions, and query length we performed 1,000 random queries and report the average result.

4.2 Evaluation

In previous work on indexing of time series, indexing schemes have been evaluated by comparing the time taken to execute a query. However this method has the disadvantage of being sensitive to the implementation of the various indexing schemes being compared. For example in [1], the authors carefully state that they use the branch and bound optimization for the Sequential-Scan (a standard indexing strawman). However, in [11] and [23] the authors do not tell us whether they are comparing their indexing schemes to optimized or unoptimized Sequential-Scan. This is a problem because the effect of the optimization can be as much as two orders of magnitude, which is far greater than the speedup reported.
As an example of the potential for implementation bias in this work consider the following. At query time *F-index* must do a Fourier transform of the query. We could use the naïve algorithm which is $O(n^2)$ or the faster radix-2 algorithm (padding the query with zeros for $n \neq 2^{integer}$ [18]) which is $O(n\log n)$. If we implemented the simple algorithm it would make our indexing method perform better relative to *F-index*.
To prevent implementation bias we will compare our indexing scheme to *F-index* by reporting the *P*, the fraction of the database that must be examined before we can guarantee that we have found the nearest match to our query.

$$P = \frac{\textit{Number of objects retreived}}{\textit{Number of objects in database}} \tag{6}$$

Note the value of P depends only on the data and the queries and is completely independent of any implementation choices, including spatial access method, page size, computer language or hardware. It is a fair evaluation metric because it corresponds to the minimum number of disk accesses the indexing scheme must make, and disk time dominates CPU time. A similar idea for evaluating indexing appears in [10].

4.3 Experimental Results

Figure 3 shows the results of comparing *PCA-index* to *F-index* on a variety of datasets. Experiments in [1,8] suggest a dimensionality of 6 for *F-index*. For completeness we tested of a range of dimensionalities, however only even numbers of dimensions are used because *F-index* (unlike *PCA-index*) is only defined for even numbers. We also tested over a variety of query lengths. Naturally, one would expect both approaches to do better with more dimensions and shorter queries, and the results generally confirm this.

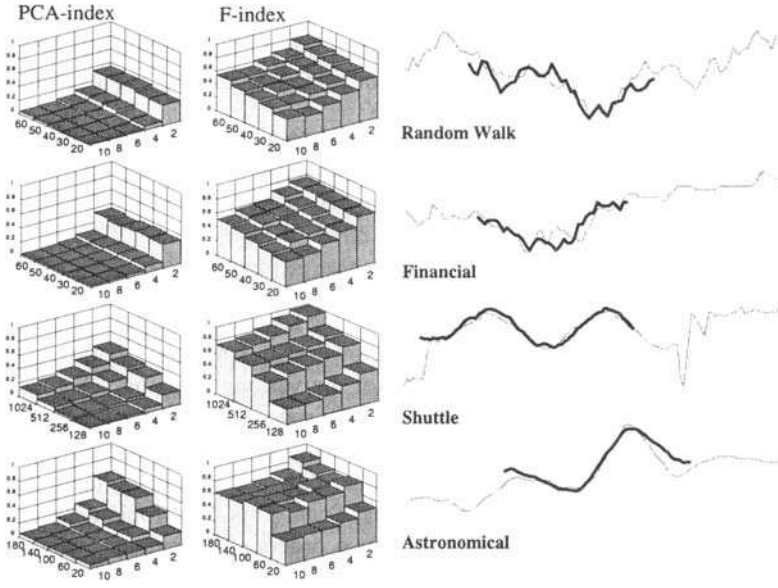


Figure 3: The fraction of the database which must be retrieved from disk using the two indexing schemes compared in this paper, together with sample queries and a section containing the corresponding best match. Each pair of 3d histograms represents a different dataset. Each bar in the 3d histogram represents P, the fraction of the database that must be retrieved from disk for a particular combination of index dimensionality and query length (averaged over 1,000 trials)

For low dimensionalities, say 2-4, *PCA-index* generally outperforms *F-index* by about a factor of two. However as the dimensionality increases the difference between the approaches grows dramatically. At a dimensionality of ten, *PCA-index* outperforms *F-index* by a factor of 81.4 (averaged over the 5 datasets in Fig 3). Competitive index

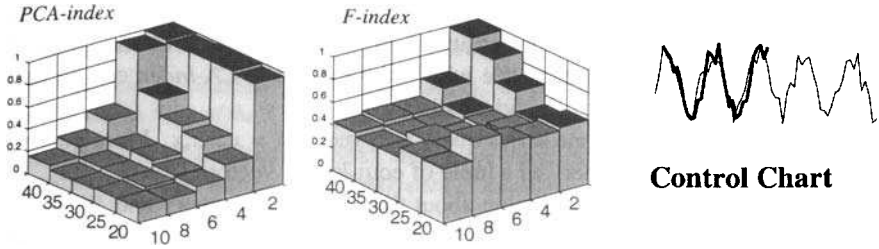


Figure 4: The result of experiments on the Control Dataset, with a sample query and a section containing the corresponding best match. The black topped 3d histogram bars indicate where *F-index* outperforms *PCA-index*

trees can easily handle dimensionalities of ten or greater [3,12].

The Control dataset shown in Fig. 4 contains the only instances where *F-index* outperformed *PCA-index,* so we will consider it in more detail. This dataset is a sine wave with noise. With just two dimensions (corresponding to the real and imaginary parts of a single Fourier coefficient) *F-index* can model a sine wave very well. In contrast, at the same dimensionality *PCA-index* has several entire periods contained within a single frame, thus all frames have approximately the same value and *PCA-index* has little discriminating power. However the situation changes dramatically as the dimensionality increases. Because most of the energy is concentrated in the first coefficient, adding more dimensions does not improve *F-index's* performance. In contrast *PCA-index* extracts great benefit from the extra dimensions. Once the frame size is less than a single period of the sine wave its performance increases dramatically.

This special case clearly illustrates a fact that can also be observed in all the other experiments, *PCA-index* is able to take advantage of extra dimensions much more that *F-index.*

5 Generalizing the Distance Measure

Although the Euclidean distance measure is optimal under several restrictive assumptions [1], there are many situations where a more flexible distance measure is desired [13]. The ability to use these different distance measures can be particularly useful for incorporating domain knowledge into the search process. One of the advantages of the indexing scheme proposed in this paper is that it can handle many different distance measures, with a variety of useful properties. In this section we will consider one very important example, weighted Euclidean distance. To the author's knowledge, this is the first time an indexing scheme for weighted Euclidean distance has been proposed.

5.1 Weighted Euclidean Distance

It is well known in the machine learning community that weighting of features can greatly increase classification accuracy [22]. In [14] we demonstrated for the first time that weighing features in time series queries can increase accuracy in time series classification problems. In addition in [13], we demonstrated that weighting features (to-

gether with a method for combining queries) allows relevance feedback in time series databases. Both [14,13] illustrate the utility of weighted Euclidean metrics, however no indexing scheme was suggested. We will now show that *PCA-index* can be easily modified to support of weighted Euclidean distance.

In Section 3.2, we denoted a time series query as a vector $X = x_1,\ldots,x_n$. More generally we can denote a time series query as a tuple of equi-length vectors $\{X = x_1,\ldots,x_n , W = w_1,\ldots,w_n\}$ where X contains information about the shape of the query and W contains the relative importance of the different parts of the shape to the query. Using this definition the Euclidean distance metric in Eq. 1 can be extended to the weighted Euclidean distance metric *DW*:

$$DW([X,W],Y)=\sqrt{\sum_{i=1}^{n} w_i(x_i-y_i)^2} \tag{7}$$

We can perform weighted Euclidean queries on our index by making two simple modifications to the algorithm outlined in Table 2. We replace the two distance measures *D* and *DR* with *DW* and *DRW* respectively. *DW* is defined in Eq. 7 and *DRW* is defined as:

$$\overline{w}_i = \min(w_{\frac{n}{N}(i-1)+1},\mathrm{K}\ ,w_{\frac{n}{N}i}), \qquad DRW([\overline{X},\overline{W}],\overline{Y})=\sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N}\overline{w}_i(\overline{x}_i-\overline{y}_i)^2} \tag{8}$$

Note that it is not possible to modify *F-index* in a similar manner, because each coefficient represents amplitude and phase of a signal that is added along the entire length of the query

6 Conclusions

We have introduced a dimensionality reduction technique that allows fast indexing of time series. We performed extensive empirical evaluation and found our method outperforms the current best known approach by one to two orders of magnitude. We have also demonstrated that our technique can support weighted Euclidean queries.

In future work we intend to further increase the speed up of our method by exploiting the similarity of adjacent sequences (in a similar spirit to the "trail indexing" technique introduced in [8]). Additionally, we hope to show the speedup obtained by *PCA-index* will support a variety of time series datamining algorithms that scale poorly to large datasets, for example the rule induction algorithm proposed in [5].

References

1. Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient similarity search in sequence databases. *Proc. of the 4th Conference on Foundations of Data Organization and Algorithms.*

2. Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in times-series databases. In *VLDB.*

3. Chakrabarti, K & Mehrotra, S. (1999). The Hybrid Tree: An Index Structure for High Dimensional Feature Spaces. *Proc of the IEEE International Conference on Data Engineering.*

4. Chan, K. & Fu, W. (1999). Efficient Time Series Matching by Wavelets. *Proceedings of the 15th International Conference on Data Engineering.*

5. Das, G., Lin, K. Mannila, H., Renganathan, G., & Smyth, P. (1998). Rule Discovery from Time Series. In *Proc of the 3rd Inter Conference of Knowledge Discovery and Data Mining.*

6. Debregeas, A. & Hebrail, G. (1998). Interactive interpretation of Kohonen maps applied to curves. *Proc of the 4th International Conference of Knowledge Discovery and Data Mining.*

7. Faloutsos, C. & Lin, K. (1995). Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD Conf.*, pp 163-174.

8. Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD Conf.*, Minneapolis.

9. Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Conf.*, pp 47-57.

10. Hellerstein, J. M., Papadimitriou, C. H., & Koutsoupias, E. (1997). Towards an Analysis of Indexing Schemes. 16th ACM SIGACT- Symposium on Principles of Database Systems.

11. Huang, Y. W., Yu, P. (1999). Adaptive Query processing for time-series data. *Proceedings of the 5th International Conference of Knowledge Discovery and Data Mining*. pp 282-286.

12. Kanth, K.V., Agrawal, D., & Singh, A. (1998). Dimensionality Reduction for Similarity Searching in Dynamic Databases. In *Proc. ACM SIGMOD Conf.*, pp. 166-176.

13. Keogh, E. & Pazzani, M. (1999). Relevance Feedback Retrieval of Time Series Data. *Proc. of the 22th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.*

14. Keogh, E., & Pazzani, M. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. *Proceedings of the 4th International Conference of Knowledge Discovery and Data Mining*. pp 239-241, AAAI Press.

15. Keogh, E., & Smyth, P. (1997). A probabilistic approach to fast pattern matching in time series databases. *Proc. of the 3rd Inter Conference of Knowledge Discovery and Data Mining*

16. Park, S., Lee, D., & Chu, W. (1999). Fast retrieval of similar subsequences in long sequence databases. *In 3rd IEEE Knowledge and Data Engineering Exchange Workshop.*

17. Scargle, J. (1998). Studies in astronomical time series analysis: v. Bayesian blocks, a new method to analyze structure in photon counting data. Astrophysical Journal, Vol. 504.

18. Shatkay, H. (1995). The Fourier Transform - a Primer, Technical Report CS-95-37, Department of Computer Science, Brown University.

19. Shatkay, H., & Zdonik, S. (1996). Approximate queries and representations for large data sequences. *Proc. 12th IEEE International Conference on Data Engineering*. pp 546-553.

20. Struzik, Z. & Siebes, A. (1999). The Haar Wavelet Transform in the time series similarity paradigm. 3rd European Conference on Principles and Practice of KDD.

21. Refiei, D., & Mendelzon, A. (1997). Similarity-Based queries for time series data. In *Proc. ACM SIGMOD Conf.*, pp. 13-25.

22. Wettschereck, D., Aha, D. & Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. AI Review, Vol 11, Issues 1-5.

23. Yi, B,K., Jagadish, H., & Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. *IEEEE International Conference on Data Engineering*. pp 201-208.

Missing Value Estimation Based on Dynamic Attribute Selection

K.C. Lee, J.S. Park, Y.S. Kim, and Y.T. Byun

Department of Computer Science
Hong-Ik University
Mapogu, Seoul, Korea 121-791
{lee,jspark,yskim,byun}@cs.hongik.ac.kr

Abstract. Raw Data used in data mining often contain missing information, which inevitably degrades the quality of the derived knowledge. In this paper, a new method of guessing missing attribute values is suggested. This method selects attributes one by one using attribute group mutual information calculated by flattening the already selected attributes. As each new attribute is added, its missing values are filled up by generating a decision tree, and the previously filled up missing values are naturally utilized. This ordered estimation of missing values is compared with some conventional methods including Lobo's ordered estimation which uses static ranking of attributes. Experimental results show that this method generates good recognition ratios in almost all domains with many missing values.

1 Introduction

Data Mining techniques have been developed to extract knowledge from a large amount of raw data. However, in the real world data, because data are gathered from many sources, some attribute values are often missing. Properly filling up the missing information may reduce the error rates of the derived knowledge. This paper introduces a new method of filling up missing values. The conventional probabilistic method used in the mining system C4.5[5] determines missing values according to the probability of the corresponding values. Lobo's method[4] fills up values of each attribute one by one according to the static ranks of their mutual information. In this paper, we present a new ordered method of estimating missing values, based on a dynamic attribute selection by considering already selected attributes. Extensive experiments show that our method much improves recognition rates in many domains, and especially works well even in environments with high missing value ratios.

This work was supported by Korea Science Foundation under contract 97-01-02-04-01-3.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 134-137, 2000.
© Springer-Verlag Berlin Heidelberg 2000

2 Related Works

The simplest approach for dealing with missing attribute values seems to ignore instances with any missing attribute values[6]. This approach cannot utilize any information other known attributes may convey, and severely reduces the number of training data especially in the case of high missing ratios. Another simple method, the majority method[2], fills up each missing attribute value with the highest frequency value of the attribute. Extensive experimentation has been conducted by Quinlan[7] including building up decision trees for each attribute to decide missing attribute values, and he adopted the probabilistic method[1] for the C4.5 data mining system[5]. Lobo's method[4] also builds decision trees for each attribute. However, the attributes are statically ordered according to their mutual information. The ordering of the attributes may be important in that the missing value estimation for an attribute may be improved by the data with attributes previously filled up.

In this paper, a new missing value estimation method is suggested. The attributes are selected one by one based on the feature group mutual information[3] calculated by flattening previously selected attributes. The corresponding decision trees are dynamically generated and previously filled up missing values play their roles.

3 Estimation of Missing Attribute Values

In filling up missing attribute values, it is desirable to utilize the information hidden in the data. Static methods like the majority method and the probabilistic method use information directly derivable from the distribution of each attribute, regardless of its relation with the other attributes. To utilize inter-attribute information, some information theoretic measures have been defined[6].

Class uncertainty may be measured by the entropy as in (1). Here C implies the set of all possible classes, and c implies a specific class.

$$H(C) = -\sum_{c=1}^{Nc} P(c) \log P(c) \qquad (1)$$

Class uncertainty after knowing the values of an attribute F may be measured by the conditional entropy as in (2). Here F implies an attribute the values of which are known, and f implies a specific attribute value of the attribute F.

$$H(C \mid F) = -\sum_{f=1}^{Nf} P(f)(\sum_{c=1}^{Nc} P(c \mid f) \log P(c \mid f) \qquad (2)$$

How much uncertainty is reduced after knowing the values of an attribute is measured by the mutual information defined by (3).

$$I(C;F) = H(C) - H(C \mid F) \qquad (3)$$

Lobo's method ranks attributes according to their mutual information defined in (3), and builds trees in that order for each attribute to decide missing values.

Once some attributes are already selected and their corresponding missing values are filled up, it seems reasonable to select the next attribute for missing value fill-up by considering its relation with the previously selected attributes. Hence we flatten the previously selected attributes into one big attribute with many values, and accordingly,

conditional class uncertainty has been defined as in (4). Here, S is the attribute generated by flattening all the already selected attributes, and (f, S) is a flattened attribute generated by merging the new attribute f with S.

$$H(C \mid (f,S)) = -\sum_{s' \in S} P(s')(\sum_{c=1}^{Nc} P(c \mid s') \log P(c \mid s')) \qquad (4)$$

S = the set of feature values of (f,S)

The attribute values of a new flattened attribute (f,S) belongs to $F \times S_1 \times S_2 \times \ldots \times S_k$, where S_i is i-th selected attribute. For example, assume that S is the flattened attribute of two already selected attributes, *shape* and *length*, such that shape may have values *rectangle* or *triangle* and *length* may have values *red* or *yellow*. S may have flattened attribute values (*rectangle*, *red*), (*rectangle*, *yellow*), (*triangle*, *red*) and (*triangle*, *yellow*). If the candidate attribute *length*(=f) may have values long or short, (f,S) may have values like (*rectangle*, *red*, *long*) and (*triangle*, *yellow*, *short*). However, any non-existing combination of attribute values are removed.

We choose each next attribute such that the feature group mutual information defined in (5) is maximum. That is, a new attribute is selected such that the attribute together with all the previously selected ones can best decide the class of unseen data.

$$GI(C;(f,S)) = H(C) - H(C \mid (f,S)) \qquad (5)$$

A new attribute f is dynamically selected such that (5) for the attribute f is maximum. For each attribute dynamically selected, a decision tree is built to fill up the missing values for the attribute, and the previously filled up attribute values may contribute to later decisions. The detailed algorithm of this method is as follows.

Step 1: AS $\leftarrow$ set of all the attributes; S $\leftarrow$ { };

Step 2: From AS, select an attribute F with max information gain ratio;

AS $\leftarrow$ AS – {F}; S $\leftarrow$ S $\cup$ {F};

Step 3: Do the following until the stopping criterion is met

Select F from FS that maximizes GI(C; (F,S));

Fill up missing values for F by building up a decision tree for F;

FS $\leftarrow$ FS – {F}; S $\leftarrow$ S $\cup$ {F};

The stopping criterion is met in step 3 if adding one or two more attributes does not improve the performance. For the performance measure, we empirically adopted pessimistic error rates to estimate the test data error rates using the training data.

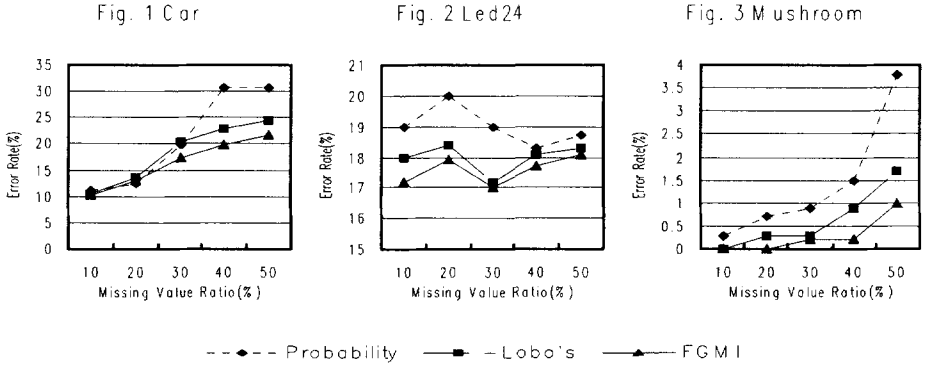
4 Experimental Results

The experimental data set are selected mainly from UCI repository[8] and summarized in Table 1 and the results are shown in Fig. 1 to 3.

Table 1. Summary of Data sets

Name	Instances	Attributes	Classes
Car	1728	6	4
Led24	3000	8	10
Mushroom	8124	22	2
Yeast	1484	7	8

Ten-fold cross validations are used and the average values are shown in the figures. The results consistently show the effectiveness of our method in most domains especially when missing value ratios are high. . In the figures this method(FGMI) is compared with Lobo's method and the probability method. 10 to 50 percent of missing values are intentionally added to each data set to test the proposed methods.



5 Concluding Remarks

A new method of filling up missing values has been suggested in this paper. The conventional methods like the majority method or the probabilistic method do not reflect the inter-attribute dependencies, and Lobo's method relies on the static ranks of the attributes. Contrarily, this method reflects the dynamic nature of the attribute selections, and much reduces error rates in most domains with high missing ratios.

References

1. B. Cestnik and et al., "Assistant-86: A Knowledge-elicitation Tool for Sophisticated Users," Progress in Machine Learning, Sigma Press, UK, 1987.
2. I. Kononenko and E. Roscar, "Experiments in Automatic Learning of Medical Diagnostic Rules," Technical Report, Jozef Stefan Institute, Yugoslavia, 1984.
3. K.C. Lee, "A Technique of Dynamic Feature Selection Using the FGMI," Lecture Notes in Artificial Intelligence 1574, pp.138-142, Springer, 1999.
4. O.O. Lobo and M. Numao, "Ordered Estimation of Missing Values," Lecture Notes in Artificial Intelligence 1574, pp.499-503, Springer, 1999.
5. J.R. Quinlan, "Unknown Attribute Values," C4.5 Programs for Machine Learning, pp.27-32, Morgan Kaufmann, 1993.
6. J.R. Quinlan, "Induction of Decision Trees," Machine Learning:1, pp.81-106, 1986.
7. J.R. Quinlan, "Unknown Attribute Values in Induction," Proc. of the 6th International Machine Learning Workshop, pp.164-168, Morgan Kaufmann, 1989.
8. http://www.ics.uci.edu/~mlearn/MLRepository.html, UCI Machine Learning Repository, Univ. of California, Dept. of Info. Computer Science, Irvine, CA, 1998.

On Association, Similarity and Dependency of Attributes

Yi Yu Yao[1] and Ning Zhong[2]

[1] Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2
E-mail: yyao@cs.uregina.ca
[2] Department of Computer Science and Systems Engineering, Faculty of Engineering
Yamaguchi University, Tokiwa-Dai, 2557, Ube 755, Japan
E-mail: zhong@ai.csse.yamaguchi-u.ac.jp

1 Introduction

Association, similarity, and dependency of attributes represent useful information and knowledge that can be derived from data sets. Similarities indicate the closeness of attributes reflected by their values on a set of objects. Two attributes are similar if every object is likely to have the same value on them. Data and functional dependencies show the connection and association between attributes. They are characterized by the problem of determining the values of one set of attributes based on the values of another set. Two levels of dependencies, referred to as the *local* and *global* dependencies, may be observed. The local dependencies show how *one* specific combination of values on one set of attributes determines *one* specific combination of values on another set. The well known association rules, which state the presence of one set of items implies the presence of another set of items, may be considered as a special kind of local dependencies. The global dependencies show *all* combinations of values on one set of attributes determine *all* combinations of values on another set of attributes. Functional dependencies in relational databases are examples of global dependencies.

The main objective of this paper is to present an outline of a systematic study on the characterization, classification, quantification, and interpretations of various types of relationships between attributes, as well as their connections to each other. For clarity, binary information tables and some particular measures are used [2, 3]. The results from the study may have significant implications in the understanding of fundamental issues of data mining in general.

2 Overview of Relationships Between Attributes

In a binary information table, a set of objects are represented by using a finite set of binary attributes taking values over the domain $\{0, 1\}$. Transaction data may be easily represented by a binary information table. For two attributes x and y, an association rule, $(x = 1) \Leftarrow (y = 1)$, states that the occurrence of y warrants the occurrence of x. Let $m(x = 1)$ denote the number of rows whose

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 138-141, 2000.
© Springer-Verlag Berlin Heidelberg 2000

value on x is 1 in the table. The confidence of an association rule is defined by:

$$conf(x_1 \Leftarrow y_1) = |m(x_1, y_1)|/|m(\cdot, y_1)|, \tag{1}$$

where x_0 and x_1 stand for the conditions $x = 0$ and $x = 1$, respectively, and hence $m(x_1, y_1) = m(x = 1, y = 1)$, and $m(\cdot, y_1) = m(y = 1)$. An association rule $x_1 \Leftarrow y_1$ does not say anything about the occurrence of y given the occurrence of x. It deals with *one-way* dependency. An association rule uses only one of the four possible combinations of co-occurrence. It thus reflects *local* data dependency. In summary, association rules summarize local one-way data dependency. A two-way association rule $x_1 \Leftrightarrow y_1$ states that the occurrence of x suggests the occurrence of y, and vice versa. The confidence of $x_1 \Leftrightarrow y_1$ is defined by [2]:

$$conf(x_1 \Leftrightarrow y_1) = |m(x_1, y_1)|/|m(x_1, \cdot)||m(\cdot, y_1)|. \tag{2}$$

In information-theoretic terms, it measure the mutual information between x_1 and y_1. It is closely related to the confidence of one-way association.

A simple similarity measure of two attributes x and y is defined by the normalized co-ordination level matching as follows:

$$s(x, y) = \frac{|m(x = 1) \cap m(y = 1)|}{|m(x = 1) \cup m(y = 1)|} = \frac{|m(x_1, y_1)|}{|m(x_1, \cdot) \cup m(\cdot, y_1)|}. \tag{3}$$

It reaches the maximum value 1 when the columns labelled by x and y are identical, and reaches the minimum value 0 when there is no co-occurrence of the $(1, 1)$ combination. The similarity measure does not depend on the combination $(0, 0)$. The co-occurrences of combination $(0, 0)$ do not increase, nor decrease, the similarity between two attributes.

For measuring global data dependencies, we use entropy related measures [1]. One-way dependency $y \Rightarrow x$ may be measured by conditional entropy:

$$H(x \mid y) = -\sum_i \sum_j \frac{|m(x_i, y_j)|}{N} \log \frac{|m(x_i, y_j)|}{|m(\cdot, y_j)|}, \tag{4}$$

where N is the number of rows in the table. Two-way dependency $x \Leftrightarrow y$ may be measured by mutual information:

$$I(x; y) = \sum_i \sum_j \frac{|m(x_i, y_j)|}{N} \log \frac{|m(x_i, y_j)|}{|m(x_i, \cdot)||m(\cdot, y_j)|}. \tag{5}$$

The mutual information measures the *divergence* of the joint distribution of x and y from the independence distribution constructed using the marginals of x and y. A larger divergence implies a higher degree of probabilistic dependency.

Functional dependency is a well established notion that summarizes logical relationship between attributes in databases. A functional dependency, $y \rightarrow x$, states the semantics constraint on the values of x and y. If $y \rightarrow x$ holds, then all rows in the table having the same value on y must have the same value on x. That is, the value of y determines the value of x. All four possible combinations of values of x and y are considered in functional dependency. A functional dependency shows global data dependency. Therefore, all local data dependencies, such as association rules and similarity, do not fully reflect functional dependency.

3 Comparisons of Various Relationships

If the functional dependency $y \rightarrow x$ holds, one may conclude that the confidence $conf(x_1 \Leftarrow y_1)$ is either 1 or 0. From the value of $conf(x_1 \Leftarrow y_1)$ we cannot conclude any functional dependency between x and y. In comparison, the confidence of two-way association $x_1 \Leftrightarrow y_1$ is related to functional dependency to a lesser degree. If $y \rightarrow x$ holds, we can say that the value of $conf(x_1 \Leftrightarrow y_1)$ is either 0 or $|m(x_1)|^{-1}$. If $s(x,y) = 1$, we can conclude $x \rightarrow y$ and $y \rightarrow x$. For $s(x,y) = 0$, we cannot infer any functional dependency between x and y. Conversely, if $y \rightarrow x$ holds, we cannot say too much about $s(x,y)$. It may happen that value of $s(x,y)$ is 0, 1, or any number between 0 and 1. If both functional dependencies $x \rightarrow y$ and $y \rightarrow x$, we have either $s(x,y) = 0$ or $s(x,y) = 1$.

Information-theoretical measures make use of all four possible combinations and are closely related to functional dependency. A functional dependency $y \rightarrow x$ holds if and only if $H(x \mid y) = 0$. In this case, x totally depends on y. If the occurrence of x is probabilistically independent of y, we have $H(x \mid y) = H(x)$, or equivalently $H(y \mid x) = H(y)$, or $H(x,y) = H(x) + H(y)$. In this case, the value of y tells nothing about the value of x, and vice versa. For the mutual information, the functional dependency $y \rightarrow x$ holds if and only if we have $I(x;y) = H(x)$. If both functional dependency $x \rightarrow y$ and $y \rightarrow x$ hold, we must have $I(x;y) = H(x) = H(y)$. If x and y are probabilistically independent, the mutual information reaches the minimum value 0.

One may consider associations derivable from other combinations. For the confidence measure, we have $conf(x_0 \Leftarrow y_1) = 1 - conf(x_1 \Leftarrow y_1)$. This connection reveals some difficulties in the interpretation of association rules. Typically, an association rule $x_1 \Leftarrow y_1$ is interpreted using an IF-THEN statement, IF y_1 THEN x_1 with confidence c. The associated meaning of the statement is that the *presence* of y warrants or suggests the *presence* of x. In the same way, one may argue that $x_0 \Leftarrow y_1$ can be paraphrased as saying that the *presence* of y warrants or suggests the *absence* of x. For a small confidence value of $x_1 \Leftarrow y_1$, e.g., < 0.5, such an interpretation is somehow counter intuitive. Similar observations can be made regarding associations rules $x_1 \Leftarrow y_1$ and $x_1 \Leftarrow y_0$. This suggests that it may not be sufficient to consider only one type of associations characterized by the combination $(1,1)$. In the calculation of confidence of $x_1 \Leftrightarrow y_1$, additional information is used. It relies on an intuitively appealing interpretation of association in terms of probabilistic independence. On the other hand, the similarity measure defines association in a different manner, although the same information is used in its calculation.

For an association rule $x_1 \Leftarrow y_1$, another measure called support is defined by $supp(x_1, y_1) = |m(x_1, y_1)|/N$. It is also the support of $x_1 \Rightarrow y_1$. The confidence and support measures are not independent. They are related to each other by: $conf(x_1 \Leftarrow y_1) = supp(x_1, y_1)/[supp(x_1, y_1) + supp(x_0, y_1)]$. Only partial co-occurrence information about x and y are used the calculation of confidence. The confidence, similarity, conditional entropy, and mutual information differ from each other by the amount of co-occurrence information used. In terms of support and confidence, the similarity measure, conditional en-

tropy, and mutual information can be expressed. The similarity measure uses co-occurrence information about three combinations except $(0,0)$. In the extreme cases, the similarity measure and the confidence measure are related by: (a). $s(x,y) = 0$ iff $conf(x_1 \Rightarrow y_1) = 0$ iff $conf(x_1 \Leftarrow y_1) = 0$, and (b). $s(x,y) = 1$ iff $conf(x_1 \Rightarrow y_1) = 1$ and $conf(x_1 \Leftarrow y_1) = 1$. For three attributes x, y and z, if $s(x,y) \geq s(x,z)$, we cannot infer $conf(x_1 \Leftarrow y_1) \geq conf(x_1 \Leftarrow z_1)$. That is, a larger similarity value does not suggest a higher level of association. Conversely, a higher level of association does not imply a larger degree of similarity. Nevertheless, when the supports of x_1 and y_1 are close, it is likely that a very strong similarity suggests a high level of association. Likewise, it is unlikely that a very weak similarity suggests a high level of association.

The conditional entropy is a kind of average of various one-way associations of values of x and y, while the mutual information is a kind of average of various two-way association. A large value of such a global measure does not necessarily warrants a large value for every one of the local associations. Like the confidence of an association rule, the the similarity measure focuses mainly on the associations characterized by the co-occurrence $(1,1)$. To a large degree, the similarity measure is reflected by one term in the mutual information, namely, $conf(x_1 \Leftrightarrow y_1)$ measuring the strength of two-way association of x_1 and y_1.

4 Conclusion

Many different forms of knowledge and information can be derived from a large data set. Relationships between attributes represent an important class. An analysis of possible relationships between attributes and their connections may play an important role in data mining. The results from our preliminary study show that at least three types of relationships can be derived. They reflect the association, similarity and dependency of attributes. Various measures have been examined for quantifying the strength of these relationships. The confidence and similarity show the local dependency, while conditional entropy and mutual information show the global dependency. Furthermore, confidence and conditional entropy represent one-way dependency, and similarity and mutual information represent two-way dependency. These measures have major difference from each other. They are also related to each other.

The results of this investigation also suggest the needs for the study of new algorithms in data mining. As future research, we will focus on data mining algorithms based on similarity measures.

References

1. Yao, Y.Y., Wong, S.K.M., and Butz, C.J. On information-theoretic measures of attribute importance, *Proceedings of PAKDD'99*, 133-137, 1999.
2. Yao, Y.Y. and Zhong, N. An analysis of quantitative measures associated with rules, *Proceedings of PAKDD'99*, 479-488, 1999.
3. Yao, Y.Y. and Zhong, N. Granular computing using information tables, manuscript, 1999.

Prototype Generation Based on Instance Filtering and Averaging

Chi-Kin Keung and Wai Lam

Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Shatin, Hong Kong
{ckkeung,wlam}@se.cuhk.edu.hk

Abstract. We propose a new algorithm, called Prototype Generation and Filtering (PGF), which combines the strength of instance-filtering and instance-averaging techniques. PGF is able to generate representative prototypes while eliminating noise and exceptions.We also introduce a distance measure incorporating the class label entropy information for the prototypes. Experiments have been conducted to compare our PGF algorithm with pure instance filtering, pure instance averaging, as well as state-of-the-art algorithms such as C4.5 and KNN. The results demonstrate that PGF can significantly reduce the size of the data while maintaining and even improving the classification performance.

1 Introduction

The classical nearest neighbor (NN) algorithm has been proved to be effective in pattern classification on different applications [8]. It stores all the training instances and requires no knowledge learning from the training set. In order to classify an unseen instance, its distance with every stored training instance is calculated and the class of its nearest instance is assigned to it. This algorithm has two main drawbacks: high computational cost on run-time classification and high storage requirement. Researchers try to solve this problem by reducing the number of stored instances. By removing non-representative and noisy instances, the classification cost and storage requirement can be reduced while maintaining or even improving the classification accuracy . Instance-filtering and instance-averaging are the two most common methods to select and generate representative instances [8]. Filtering techniques try to filter out non-representative instances from original training set and averaging techniques attempt to find the most common characteristics of similar instances by merging and summarizing them. The two methods have their own strength and weaknesses. Our goal in this paper is to integrate the two methods to select and generate representative instances and eliminate noise and exceptions. To this end, we propose a new algorithm, called Prototype Generation and Filtering (PGF) which combines the strength of the two methods.

Many researchers have worked on the selection of representative instances using different methods. For examples, Hart proposes a Condensed Nearest Neighbor (CNN) which is probably the earliest method to select representative instances [14]. CNN starts by randomly storing one instance for each class as the

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 142–152, 2000.
© Springer-Verlag Berlin Heidelberg 2000

initial subset and stores instances misclassified by the current subset. A top-down variant of CNN, called Reduced Nearest Neighbor (RNN) is proposed by Gates which removes instance if the removal does not cause any misclassification of other instances [12]. Later, a number of variants of CNN have been proposed including [13,19]. The Edited Nearest Neighbor (ENN) algorithm proposed by Wilson eliminates instances misclassified by their k-nearest neighbors [21]. This algorithm retains central points since they are usually correctly classified by their k-nearest neighbors. Aha et al. introduce the well-know IB2 and IB3 algorithm which is based on CNN storing misclassified instances [2]. IB2 is similar to CNN except that instances are normalized by the range of attributes and missing value are tackled while IB3 only accepts instances with a relatively high classification accuracy compared with the frequency of the observed class. The two algorithms provide noise tolerance. Aha et al. also propose three instance pruning techniques which can determine the relative attribute relevance and handle novel attributes [1]. Zhang introduces Typical Instance-Based Learning (TIBL) which stores typical instance in the region centers [23]. Wilson and Martinez introduce three instance pruning techniques, called RT1-RT3, which removes an instance by considering its *associates*, instances in the current selected instance set having it as one of their k-nearest neighbors [22]. RT1 removes an instance if most of its associates are correctly classified without it. RT2 considers associates in the entire data set rather than those in the selected set and also sorts the instances by the distance to their nearest neighbors. RT3 is similar to RT2 except that it employs ENN to filter out noise first.

Chang's method learns representative instances by merging similar ones. It iteratively merges two closest instances and summarized them by taking the weighted average of them [6]. Bradshaw introduces the *Disjunctive Spanning* (DS) which merges instances with the ones they can correctly classified [5]. Kibler and Aha improve DS by using an *adaptive threshold* to limit the distance between two merged instances [15]. An algorithm called Nested Generalized Exemplar (NGE) is proposed by Salzberg which stores instances as hyperrectangles [18]. Wettschereck combine the NGE with KNN to form a hybrid algorithm [20].

Datta and Kibler introduce the Prototype Learner (PL) which learns artificial instances for each class by generalization of representative instances [9] in nominal domains. Then they propose the Symbolic Nearest Mean Classifiers (SNMC) [10,11] which attempts to learn a single prototype for each class using a modified Value Difference Metric proposed by Cost and Salzberg to weight symbolic features [7]. SNMC uses k-means clustering to group instances of the same class and create artificial instances using cluster means. Bezdek at el. modify Chang's method (MCA) which averages instances using simple mean and merges instances of the same class only [4].

2 Our Proposed Algorithm

2.1 Motivation

Many previous works try to find representative instances by either removing non-representative and noisy instances (instance-filtering) or merging and summarizing similar instances (instance-averaging). Indeed, the two methods have their own strength and weaknesses. Filtering methods are more flexible. We can decide different filtering rules to retain different instances such as border or central points and to eliminate noise and outliers. Besides, a consistent subset can be ensured using some filtering rules [14]. However, filtering methods cannot find the most representative instances if they are not in the original data set. They do not generalize instances so that a satisfactory data reduction is usually not gained. Some filtering methods are sensitive to noise and the order of data presentation. On the other hand, instance-averaging methods try to generate representative prototypes by generalizing common characteristics of instances. They usually have great data reduction rate by summarizing instances instead of simply selecting them. Also, noise can be effectively generalized away by merging similar instances. However, non-prototypical instances may be formed if distant instances are merged. The prototype set may even contain misclassified instances thus degrading classification accuracy.

It seems that the two methods can benefit each other. For example, data reduction in filtering methods can be improved by merging similar instances. Some noisy instances can be generalized away by averaging methods. The most representative instances are sometimes not found in the original data set by filtering rules. Therefore, more representative instances may be found if the commonest features of instances are generalized using averaging methods. On the other hand, non-prototypical instances will not be likely formed in instance-averaging techniques if outliers are eliminated before. Some filtering rules can also eliminate noise effectively so that the classification accuracy of instance-averaging methods can be greatly improved. Besides, filtering rules can also be designed to further reduce the data set size.

In view of the above motivation, we propose a framework, called *Prototype Generation and Filtering* (PGF), which combines the strength of instance averaging and instance filtering methods to generate high quality prototypes. Our objective is to significantly reduce the data set via prototypes while maintaining the same level of, or even better classification performance.

2.2 The Framework of Our Approach

PGF consists of an instance-averaging and an instance-filtering components. We first describe the averaging method and then the filtering method. Then we present how the two methods are integrated to gain high quality prototypes.

Instance-Averaging Component. Instance-averaging methods attempt to find the commonest features of instances in the same class by merging similar instances. They intend to generate ideally one prototype for each class to

classify unseen cases. Many instance-averaging methods make use of clustering techniques. Some methods adopt k-means clustering [10] and others apply hierarchical methods [6]. Using k-means clustering, we have to decide the number of prototypes (clusters) beforehand. Researchers try different values of k to find the optimal results. In hierarchical clustering methods, the number of prototypes can be determined by stopping rules. Since predefining the number of prototypes will greatly restrict the search space for prototype selection, we adopt hierarchical clustering method in our proposed framework. The quality of prototypes can be evaluated by an evaluation function. Figure 1 shows the algorithm of the instance-averaging component used in our PGF algorithm.

1. Let prototype set = *Training Set.*
2. Merge two nearest prototypes in current prototype set.
3. Evaluate current prototype set.
4. Repeat 2 and 3 until no. of prototype = no. of class.
5. Output prototype set with maximum evaluation score.

Fig. 1. Instance-averaging component using hierarchical clustering method

A prototype essentially contains a set of original instances. It is represented by the mean of the constituent instances in the prototype. As shown in Figure 1, each instance is considered as a prototype initially. At each iteration, two prototypes with shortest distance in the current prototype set are merged to form a new prototype. The new prototype is the mean of all instances contained in the merged prototypes and the majority class of the contained instances becomes the class of the prototypes. The new prototype set is then evaluated by an evaluation function to predict the quality of the prototype set. The best prototype set is stored. These merging and evaluation processes continue until the number of prototypes reduces to the number of classes in the data set. Finally, the prototype set with optimal evaluation score is the output.

As our objective is to learn prototypes to classify unseen cases, classification accuracy on unseen instances is a good evaluation criterion to evaluate the quality of prototypes. We divide the training data into a sub-training set and a tuning set. Prototypes are learned using the sub-training set while the tuning set is used to evaluate the learned prototypes.

In order to generate homogeneous prototypes, many instance-averaging methods only merge instances of the same class. For example, [10] partitions the original data set by class and merge instances within each partition. Despite the homogeneity of the prototypes formed, these methods may distort the data distribution. Besides, the strength of averaging method to generalize away mislabeled instances of other labels in compact regions is disabled. In view of this, we design a distance measure which considers both the Euclidean distance and

the entropy of the merged prototypes. The entropy, $Ent(\mathbf{x})$, of a prototype $\boldsymbol{x}$ is related to the class distribution of the instances contained in the prototype. It is defined as:

$$Ent(\boldsymbol{x}) = -\sum_{i=1}^{c} R(\mathbf{x}, i) \log R(\mathbf{x}, i)$$

where $R(\mathbf{x}, i)$ is the relative frequency of the occurrence of the class label i in the prototype $\boldsymbol{x}$. When two prototypes $\boldsymbol{x}$ and $\boldsymbol{y}$ are considered to merge, the entropy distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as:

$$E(\boldsymbol{x}, \boldsymbol{y}) = Ent(\boldsymbol{z})$$

where $\boldsymbol{z}$ is a hypothetic prototype generated by merging $\boldsymbol{x}$ and $\boldsymbol{y}$. If a low entropy is calculated, most instances in the merged prototypes are of the same class. Therefore, entropy encourages homogeneous prototypes to be merged.

The Euclidean distance of two prototypes is normalized to value from 0 to 1 which is of the same range of entropy. The normalized Euclidean distance of two prototypes $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as:

$$D(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sqrt{\sum_{i=1}^{n}(x_i - y_i)}}{\sqrt{\sum_{i=1}^{n}(\max_i - \min_i)}}$$

where n is the number of attributes, x_i, y_i are the i-th attribute values of $\boldsymbol{x}$ and $\boldsymbol{y}$ and $\max_i$, $\min_i$ are the maximum and minimum value of the i-th attribute in the training set respectively. After the two components are calculated, a parameter α ($0 \leq \alpha \leq 1$) is then used to weight their contributions. The distance function EE of PGF is:

$$EE(\boldsymbol{x}, \boldsymbol{y}) = \alpha D(\boldsymbol{x}, \boldsymbol{y}) + (1 - \alpha) E(\boldsymbol{x}, \boldsymbol{y})$$

This distance measure favours the merging of homogeneous instances while preserving the original data distribution.

Instance-Filtering Component. As described above, instance-averaging methods cannot handle outliers and exceptions effectively. Therefore, we can design filtering rules which eliminate outliers and exceptions to solve the problem. Besides, non-prototypical and misclassified prototypes may be generated after instance-averaging. Filtering rules can help if they can discard non-representative and noisy instances. For this reason, filtering rules which can eliminate outliers, exception and noise would benefit to instance-averaging methods. We have tried two filtering methods. The first one is the ENN method introduced by Wilson [21]. This method discards instances which are misclassified by their k nearest-neighbors. As outliers and noise are seldom classified correctly by their k nearest-neighbors, they will usually be removed.

The second instance-filtering method considers the classification performance of each prototype in the prototype set. After the merging process, artificial prototypes are generated by the sub-training set. These prototypes can be used to classify the original training set and the classification accuracy of each prototype are calculated. Prototypes with accuracy lower than a certain threshold

Q will then be discarded. This method can effectively remove noisy and non-prototypical prototypes as they usually have low accuracy. Besides, outliers and exceptions can also be eliminated.

Integration of Filtering and Averaging. Our PGF algorithm integrates instance-filtering techniques in the instance-averaging component described above. Figure 2 shows the algorithm of PGF. The filtering processing is indicated by statement 4. Filtering is conducted when a new prototype set is formed after each merging process in the instance-averaging component. The filtered prototype set is then evaluated by the prototype set evaluation function and the one with optimal evaluation score will be output. After the prototype set is learned, it is used to classify unseen cases. The simple Euclidean distances between the unseen case and all the prototypes are calculated to find the nearest prototype. The majority class of its nearest prototype is then assigned to the unseen case.

1. Let prototype set = *Training Set.*
2. Merge two nearest prototypes in current prototype set.
3. Let *temp* = current prototype set.
4. Filter *temp.*
5. Evaluate *temp.*
6. Repeat 2, 3, 4 and 5 until no. of prototype = no. of class.
7. Output *temp* with maximum evaluation score.

Fig. 2. The algorithm of the PGF algorithm

3 Empirical Evaluation

3.1 Experimental Setup

We have conducted a series of experiments to investigate the performance of our PGF framework. Nineteen real-world data sets from the widely used UCI Repository [16] were tested in the experiments. For each data set, we randomly partitioned the data into ten portions. Ten trials derived from 10-fold cross-validation were conducted for every set of experiments. The mean of the storage requirement and classification accuracy of the ten trials were calculated to measure the performance for a particular data set. Attribute values are normalized by the range of features in the training set and missing values were replaced by the mean value of the feature. We first investigate the merit of our framework as compared to pure instance filtering and pure prototype generation methods. Then we compare our algorithm with existing learning algorithms, namely, C4.5 and KNN. We finally investigate the performance of our distance measure *EE* by comparing *EE* with and without the entropy consideration.

3.2 Experimental Results

The first set of experiments studies the benefits of the integration of prototype generation and prototype filtering methods. We compare two variants of our PGF algorithms with pure instance-filtering and pure instance-averaging algorithms. The first PGF uses ENN to filter prototype set (PGF-ENN) and the other one filters out prototypes with classification accuracy lower than a threshold Q (PGF-ACC). The pure instance-filtering algorithm is the simple ENN (pure-ENN) while the pure instance-averaging algorithms is the one using classification accuracy (pure-ACC) as the prototype set evaluation function. We have tried a range of parameters in algorithms compared and select the best one to represent each of them. The parameter α in our distance measure is also tuned and the best result is reported. Table 1 and Table 2 show the average storage requirement and classification accuracy of the four methods respectively.

We first compare pure-ENN with PGF-ENN. From the results in Table 2, it is found that the pure-ENN have a slightly higher performance in classification accuracy. However, the storage requirement of PGF-ENN is significantly lower than that of pure-ENN. About 57% of the training data are further reduced using PGF-ENN instead of pure-ENN. This supports the fact that instance-averaging methods can further improve data reduction rate of instance-filtering techniques by generalizing similar instances without great degradation in classification accuracy. PGF-ENN even gains a slightly increase in classification accuracy on some data sets using much fewer number of prototypes.

However, when comparing the pure instance-averaging method (pure-ACC) with PGF-ENN, we find that data reduction rate degrades using the integrated method. pure-ACC requires a 8% higher average storage requirement than PGF-ENN does. As for classification accuracy, the two algorithms gain similar results on all the data sets. The poor performance of ENN in the integrated method can be explained as below. As mentioned above, ENN discards those instances which are misclassified by their k nearest neighbors. Central instances will be retained as they are usually correctly classified by their k nearest neighbors. However, our instance-averaging method attempts to generalize similar instances by taking the mean of the merged instances as artificial prototypes. These artificial prototypes are obviously central points which are usually retained by ENN. Also, with the influence of ENN, PGF-ENN finds its optimal prototype set in an earlier merging iteration. Therefore, its data reduction rate degrades even with the integration of ENN to filter the prototype set.

We then compare pure-ACC with PGF-ACC. Table 1 and Table 2 show that PGF-ACC gains significant improvement in average storage requirement while maintaining a high level of classification accuracy. PGF-ACC uses 11.5% fewer of the total instances to achieve a slightly lower (1.1%) classification accuracy compared with pure averaging method. PGF-ACC discards less representative prototypes by using classification accuracy as filtering rule. This accounts for the improvement of data reduction in the integrated method.

Table 1. The average storage requirement of ten trials for pure-ENN, pure-ACC, PGF-ENN and PGF-ACC

Data Set	pure-ENN	pure-ACC	PGF-ENN	PGF-ACC
balance-scale	0.827	0.162	0.239	0.022
breast-cancer-w	0.961	0.075	0.050	0.046
glass	0.701	0.097	0.215	0.052
ionosphere	0.842	0.245	0.195	0.041
iris	0.950	0.103	0.295	0.081
letter	0.755	0.591	0.546	0.196
liver-disorders	0.639	0.168	0.275	0.070
new-thyroid	0.949	0.095	0.231	0.058
optdigit	0.965	0.491	0.436	0.137
pendigit	0.983	0.238	0.261	0.113
pima-diabetes	0.734	0.018	0.013	0.011
segmentation	0.950	0.355	0.462	0.165
shuttle	0.979	0.274	0.330	0.121
sonar	0.852	0.363	0.362	0.079
vowel	0.964	0.256	0.753	0.212
wdbc	0.966	0.232	0.248	0.095
wine	0.956	0.085	0.126	0.054
wpbc	0.747	0.018	0.049	0.021
yeast	0.552	0.076	0.379	0.108
Average	0.856	0.207	0.287	0.088

In conclusion, we find that the performance of PGF depends on the choice of the filtering method. To make use of the advantage of the integrated method, the averaging and filtering techniques should target on different instances.

In the second set of experiments, we compare PGF with C4.5 and KNN. The PGF used in the comparison is PGF integrated with prototype filtering method using accuracy (PGF-ACC). In KNN, we have conducted different values of k (k=1,3,5,7,9,11,15,20) and the best k, which is 3, is used for comparison. For PGF-ACC, we have also tried a range of threshold Q in the filtering component. Table 1 depicts the size of the reduced data set of PGF (PGF-ACC). It shows that PGF only retains an average of 8.8% of the total instances while maintaining a high classification accuracy.

Table 3 shows the classification accuracy and standard deviation of the three algorithms on all the data sets. It shows that the PGF algorithm is slightly better than C4.5 in the average classification accuracy across most of the data sets and KNN has a slightly higher average classification accuracy than PGF. Using a t-test at 0.05 significant level, we find that PGF outperforms C4.5 in 5 of the data sets and has the same performance on the remaining ones. Compared with KNN, PGF performs equally well in 15 of the 19 data sets. In conclusion, PGF can achieve comparable classification performance with state-of-the-art learning algorithms such as C4.5 and KNN. More importantly PGF can drastically reduce the data size to only 8.8% of the original size on average.

Table 2. The average classification accuracy of ten trials for pure-ENN, pure-ACC, PGF-ENN and PGF-ACC. The standard deviation is given inside the bracket

Data Set	pure-ENN	pure-ACC	PGF-ENN	PGF-ACC
balance-scale	0.856 (0.034)	0.806 (0.099)	0.854 (0.050)	0.851 (0.041)
breast-cancer-w	0.971 (0.039)	0.954 (0.028)	0.947 (0.031)	0.957 (0.033)
glass	0.658 (0.205)	0.603 (0.137)	0.658 (0.261)	0.659 (0.135)
ionosphere	0.832 (0.088)	0.906 (0.046)	0.860 (0.095)	0.883 (0.070)
iris	0.953 (0.046)	0.933 (0.050)	0.933 (0.097)	0.947 (0.063)
letter	0.724 (0.036)	0.767 (0.064)	0.692 (0.046)	0.665 (0.048)
liver-disorders	0.626 (0.069)	0.597 (0.063)	0.637 (0.080)	0.591 (0.071)
new-thyroid	0.944 (0.053)	0.921 (0.074)	0.921 (0.055)	0.921 (0.049)
optdigit	0.949 (0.032)	0.946 (0.037)	0.941 (0.043)	0.923 (0.027)
pendigit	0.982 (0.016)	0.969 (0.013)	0.965 (0.014)	0.961 (0.018)
pima-diabetes	0.751 (0.100)	0.709 (0.079)	0.716 (0.111)	0.716 (0.064)
segmentation	0.943 (0.026)	0.959 (0.011)	0.932 (0.033)	0.935 (0.021)
shuttle	0.971 (0.051)	0.974 (0.046)	0.971 (0.047)	0.971 (0.050)
sonar	0.827 (0.199)	0.832 (0.072)	0.802 (0.085)	0.789 (0.100)
vowel	0.968 (0.061)	0.972 (0.027)	0.931 (0.039)	0.940 (0.047)
wdbc	0.958 (0.028)	0.940 (0.056)	0.945 (0.040)	0.933 (0.033)
wine	0.948 (0.101)	0.960 (0.046)	0.966 (0.100)	0.960 (0.046)
wpbc	0.783 (0.153)	0.768 (0.132)	0.742 (0.126)	0.737 (0.159)
yeast	0.569 (0.061)	0.561 (0.056)	0.569 (0.027)	0.520 (0.057)
Average	0.853	0.846	0.841	0.835

Table 3. The average classification accuracy of ten trials for C4.5, KNN and PGF-ACC. The standard deviation is given inside the bracket

Data Set	C4.5	KNN	PGF (PGF-ACC)
balance-scale	0.792 (0.066)	0.818 (0.052)	0.851 (0.041)
breast-cancer-w	0.939 (0.041)	0.964 (0.025)	0.957 (0.033)
glass	0.666 (0.083)	0.709 (0.199)	0.659 (0.135)
ionosphere	0.900 (0.032)	0.843 (0.049)	0.883 (0.070)
iris	0.953 (0.063)	0.940 (0.050)	0.947 (0.063)
letter	0.692 (0.043)	0.751 (0.029)	0.665 (0.048)
liver-disorders	0.642 (0.054)	0.663 (0.099)	0.591 (0.071)
new-thyroid	0.921 (0.081)	0.953 (0.063)	0.921 (0.049)
optdigit	0.824 (0.029)	0.951 (0.049)	0.923 (0.027)
pendigit	0.914 (0.015)	0.984 (0.014)	0.961 (0.018)
pima-diabetes	0.694 (0.085)	0.745 (0.076)	0.716 (0.064)
segmentation	0.951 (0.015)	0.951 (0.008)	0.935 (0.021)
shuttle	0.989 (0.045)	0.974 (0.048)	0.971 (0.050)
sonar	0.706 (0.094)	0.856 (0.108)	0.789 (0.100)
vowel	0.779 (0.046)	0.967 (0.022)	0.940 (0.047)
wdbc	0.944 (0.031)	0.965 (0.018)	0.933 (0.033)
wine	0.888 (0.081)	0.960 (0.056)	0.960 (0.046)
wpbc	0.676 (0.168)	0.742 (0.163)	0.737 (0.159)
yeast	0.545 (0.049)	0.554 (0.038)	0.520 (0.057)
Average	0.811	0.857	0.835

Table 4. The average classification accuracy and storage requirement of ten trials of 19 data sets for pure-ACC, PGF-ENN and PGF-ACC with and without entropy in distance measure

	pure-ACC		PGF-ENN		PGF-ACC	
Distance Measure	accuracy	storage	accuracy	storage	accuracy	storage
with entropy	0.846	0.207	0.841	0.287	0.835	0.088
without entropy	0.839	0.296	0.839	0.307	0.813	0.115

In the last set of experiments, we investigate the contribution of adding entropy in our distance measure. We compare our distance measure *EE*, with and without entropy. Note that if no entropy is used, the distance measure is essentially the normalized Euclidean distance. Table 4 shows the average classification accuracy and storage requirement of the pure instance-average (pure-ACC), PGF-ENN and PGF-ACC of the 19 data sets. We find that entropy has different impacts on different algorithms. For pure-ACC, storage requirement is improved from 29.6% to 20.7% with almost the same performance on classification accuracy. However, in PGF-ENN, entropy only improves both classification accuracy and storage requirement very slightly. A greater improvement in classification accuracy and data reduction rate can be found in PGF-ACC. With the addition of entropy in distance measure, homogeneous prototypes are more likely to be merged. The results show that this can help in generating more representative prototypes and a higher classification accuracy and data reduction can be gained.

4 Conclusions

We have presented a new prototype generation method, called Prototype Generation and Filtering (PGF), which integrates the strength of instance-filtering and instance-averaging techniques. We compare the data reduction rate and the classification performance with pure filtering, pure averaging, as well as C4.5 and KNN on 19 real data sets. PGF is found to be effective in reducing the data set size while maintaining or even improving the classification accuracy. PGF can so far deal with real attributes only. In the future, we intend to extend it to symbolic features by using other distance metric which is capable of measuring distance of discrete values. Different instance averaging methods can also be investigated to see its effects on the PGF algorithm.

References

1. Aha, D.W.: Tolerating Noisy, Irrelevant, and Novel Attributes in Instance-Based Learning Algorithms. International Journal of Man-Machine Studies, Vol. 36. (1992) 267–287
2. Aha, D.W., Kibler, D. and Albert, M.K.: Instance-Based Learning Algorithms. Machine Learning, Vol. 6. (1991) 37-66

3. Bareiss, R.: Exemplar-Based Knowledge Acquisition. A Unified Approach to Concept Representation, Classification, and Learning. Perspective in Artificial Intelligence, Vol 2. Academic Press (1989)
4. Bezdek, J.C., Reichherzer, T.R., Lim, G.S. and Attikiouzel, Y.: Multiple-Prototype Classifier Design. IEEE Transactions on Systems, Man, and Cyberneics, Vol. 28, no. 1. (1998) 67–79
5. Bradshaw, G.: Learning about Speech Sounds: The NEXUS project. Proceedings of the Fourth International Workshop on Machine Learning. (1987) 1–11
6. Chang, C.L.: Finding Prototypes for Nearest Neighbor Classifiers. IEEE Transactions on Computers, Vol. 23, no. 11. (1974) 1179–1184
7. Cost, S and Salzberg, S.: A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Feature. Machine Learning, Vol. 10. (1993) 57–78
8. Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. Los Alamito, CA: IEEE Computer Society Press (1991)
9. Datta, P. and Kibler, D.: Learning Prototypical Concept Description. Proceedings of the Twelfth International Conference on Machine Learning. (1995) 158–166
10. Datta, P. and Kibler, D.: Symbolic Nearest Mean Classifier. Proceedings of the Fourteenth National Conference of Artificial Intelligence. (1997) 82–87
11. Datta, P. and Kibler, D.: Learning Symbolic Prototypes. Proceedings of the Fourteenth International Conference on Machine Learning. (1997) 75–82
12. Gates, G.W.: The Reduced Nearest Neighbor Rule. IEEE Transactions on Information Theory, Vol. 18, no. 3. (1972) 431–433
13. Gowda, K.C. and Krisha, G.: The Condensed Nearest Neighbor Rule Using the Concept of Mutual Nearest Neighborhood. IEEE Transactions on Information Theory, Vol. 25, no. 4. (1979) 488–490
14. Hart, P.E.: The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory, Vol. 14, no. 3. (1968) 515–516
15. Kibler, D. and Aha, D.W.: Comparing Instance-Averaging with Instance-Filtering Learning Algorithms. Proceedings of the Third European Working Session on Learning. (1988) 63–80
16. Murphy P.M. and Aha, D.W.: UCI Repository of Machine Learning Database. Irvine, CA: University of California Irvine, Department of Information and Computer Science. http://www.ics.uci.edu/ mlearn/MLRepository.html (1994)
17. Ritter, G.L, Woodruff, H.B. and Lowry, S.R.: An Algorithm for a Selective Nearest Neighbor Decision Rule. IEEE Transactions on Information Theory, Vol. 21, no. 6. (1975) 665–669
18. Salzberg, S.: A Nearest Hyperrectangle Learning Method. Machine Learning, Vol. 6. (1991) 251-276
19. Ullmann, J.R.: Automatic Selection of Reference Data for Use in a Nearest Neighbor Method of Pattern Classification. IEEE Transactions on Information Theory, Vol. 20, no. 4. (1974) 431–433
20. Wettschereck, D.: A Hybrid Nearest–Neighbor and Nearest–Hyperrectangle Algorithm. Proceedings of the Seventh European Conference on Machine Learning. (1994) 323–335
21. Wilson, D.L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Cyberneics, Vol. 2, no. 3. (1972) 431–433
22. Wilson, D.R. and Martinez T.R.: Instance Pruning Techniques. Proceedings of the Fourteenth International Conference on Machine Learning. (1997) 403–411
23. Zhang, J.: Selecting Typical Instances in Instance-Based Learning. Proceedings of International Conference on Machine Learning. (1992) 470–479

A Visual Method of Cluster Validation with Fastmap

Zhexue Huang[1] and Tao Lin[2]

[1] E-Business Technology Institute, The University of Hong Kong, Pokfulam, Hong Kong, CHINA
[2] CSIRO Mathematical and Information Science, GPO Box 664, Canberra, ACT 2601, AUSTRALIA
tao.lin@cmis.csiro.au

Abstract. This paper presents a visual method of cluster validation using the Fastmap algorithm. Two problems are tackled with Fastmap in the interactive process of discovering interesting clusters from real world databases. That is, (1) to verify separations of clusters created by a clustering algorithm and (2) to determine the number of clusters to be produced. They are achieved through projecting objects and clusters by Fastmap to the 2D space and visually examining the results by humans. We use a real example to show how this method has been used in discovering interesting clusters from a real data set.

Key Words: Data mining, Clustering, Cluster validation, Cluster visualization

1 Introduction

Clustering data in databases is an important task in real applications of data mining techniques. A typical example is customer segmentation. In database marketing, for instance, sound customer segmentation is a necessary condition for conducting effective marketing campaigns. In telecommunication service, customer segmentation is critical in identifying potential churners and deciding proper offers to retain them. However, clustering a large real world database is by no means a trivial task to data miners because of the size and complexity of data.

A notorious characteristic of clustering is that different clustering algorithms often impose different clustering structures on data [Jain88]. Unless synthetic data with known cluster distributions are used, it is difficult to compare the clustering results of real data from different clustering algorithms. Objects in real world databases are usually represented in high dimensions and the distributions of objects are often very sparse. Outliers, noise and clusters at different levels coexist. To discover interesting clusters from these databases, an interactive approach to stepwise clustering data and validating clusters is proved effective.

In this paper, we demonstrate the effectiveness of the interactive approach to discovering hierarchically structured clusters in data. We present a visual method to use Fastmap and other techniques to visually validate clusters in the interactive process of clustering. Fastmap is a fast algorithm to project high dimensional data onto low dimensional spaces, due to Faloutsos and Lin [Falo95]. In the clustering

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 153-164, 2000.
© Springer-Verlag Berlin Heidelberg 2000

process, we use Fastmap to solve two problems, (1) to verify the separations of clusters created by a clustering algorithm and (2) to determine the number of clusters to be produced.

The interactive process of clustering is conducted as follows. Starting from a given data set, we first use Fastmap to project all objects onto a 2D space and visually determine the number of clusters to be produced. Then, we use a clustering algorithm to generate these clusters. After that, we use Fastmap again to project the clusters onto a 2D space and visualize them in different colors and/or symbols. If we see any separate clusters, we identify them as isolated clusters in the original high dimensional space. This is because a cluster separate from others in the 2D space is also separate from others in the original high dimensional space. For other clusters that overlap on the 2D display, we use other techniques to investigate whether they are separate or not. For the clusters whose separations are uncertain, we merge them into a composite cluster and use the clustering algorithm to further partition it. We interactively repeat this process to create a tree of clusters from which interesting clusters can be gradually identified. In the paper, we use a real example to show how this method has been used in discovering interesting clusters from a complex data set in a real world application.

Projection of high dimensional data onto low dimensional spaces for clustering is a common approach in cluster analysis. Fastmap was primarily designed for this purpose [Falo95]. Other widely used methods include *principal component analysis* (PCA), multidimensional scaling (MDS) [Youn87] and dimensionality reduction techniques such as *K-L* transform [Fuku90]. Ganti et al. [Gant99] has recently integrated Fastmap with the BIRCH clustering algorithm [Zhan97] to cluster data in arbitrary spaces. In their approach, Fastmap is used to project data in an arbitrary space onto a projected space in which clustering is performed. However, performing clustering in the projection space cannot guarantee the discovery of clusters in the original space. In our approach, we create clusters from the original high dimensional space and use Fastmap to validate these clusters in the projected low dimensional space. When a cluster is validated, we are able to conclude that it is a cluster in the original space.

In this work, we have chosen the *k*-prototypes algorithm for clustering high dimensional data [Huan98] because of its natural handling of categorical data. Other clustering algorithms, such as CLIQUE [Agra88], CLARANS [NgHa96], BIRCH [Zhan97], and DBSCAN [Este98], can be used as alternatives in our approach. However, these algorithms usually deal with numeric data. Data conversion is needed when categorical data is involved.

2 Building a Cluster Tree

Our approach to clustering a large data set is to use a clustering algorithm and various cluster validation methods, including Fastmap, to interactively build a tree of clusters from which interesting clusters can be discovered. In this section, we define a cluster tree and discuss how it is interactively built from a data set.

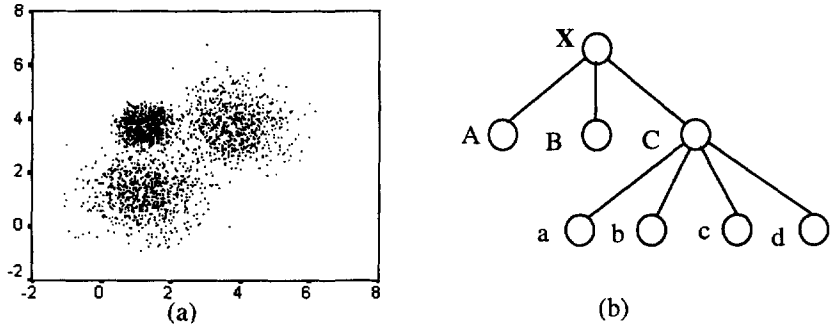


Fig. 1. (a) A data set contains six normally distributed clusters. (b) The cluster tree of the data set.

2.1 Definitions

Let X be a data set containing N objects, that is, $X = \{x_1, x_2, \ldots, x_N\}$.

Definition 1 [Theo99]: An m-clustering of X is a partition of X into m subsets (clusters), $C_1, \ldots, C_m$, which satisfies:

- $C_i \neq \varnothing,\ i = 1, \ldots, m$
- $\cup_{i=1}^{m} C_i = X$
- $C_i \cap C_j = \varnothing,\ i \neq j,\ i, j = 1, \ldots, m$

In addition, the objects in a cluster C_i are "more similar" to each other and "less similar" to objects of other clusters.

Definition 2: A clustering S with k clusters is said to be nested in a clustering T, which contains r ($< k$) clusters, if for any cluster C_1 in S, there is a cluster C_2 in T such that $C_2 \supseteq C_1$. And there exists at least one cluster in S, which holds $C_2 \supset C_1$ and $C_2 \neq C_1$.

Definition 3: A cluster tree is a sequence of nested clusterings, $\{S_0, S_1, \ldots, S_m\}$, so that for any i, j with $i < j$ and for any $C_j \in S_j$, there is $C_i \in S_i$ such that $C_i \supseteq C_j$.

Fig. 1 (a) shows a data set containing six clusters. At the high level, clusters A and B are two clusters with normal distributions while cluster C consists of four sub clusters which also have normal distributions. The cluster tree of the data set is represented in Fig. 1 (b). It consists of two clusterings of X, i.e., S_1={A, B, C} and S_2={A, B, a, b, c, d}, and S_2 is nested in S_1. We call C a composite cluster and others atomic clusters.

2.2 Discovery of a Cluster Tree

Given the data set in Fig. 1 (a) that has six clusters, we can use a clustering algorithm such as k-means to partition it into six clusters. Then we compare the clustering

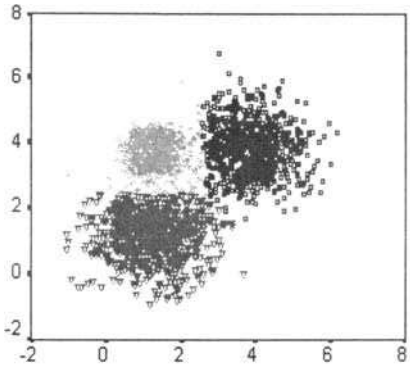


Fig. 2. Discovery of the three high level clusters in Step 1.

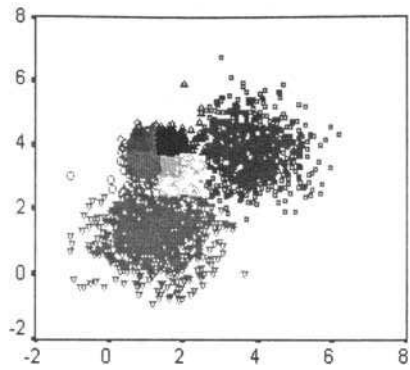


Fig. 3. Six clusters discovered in two steps by applying the *k*-means algorithm to the data.

results with the original clusters to see if the original clusters can be recovered. However, in our experiment, such an approach was not successful. On one hand, it wrongly divided clusters A and B. On the other hand it was unable to separate sub clusters in cluster C. We also tried the hierarchical clustering methods which also produced incorrect clusterings. This implies that if clusters exist at different density levels in data, it is hard to discover them in a single clustering.

To solve this problem, we took two steps to cluster the data. Again, we used *k*-means. First, we partitioned the data into three clusters. The result is shown in Fig. 2. In this step, we successfully recovered the three high level clusters A, B and C. Then we used *k*-means to partition only cluster C into four clusters. In this step, we successfully recovered the four sub clusters a, b, c, and d. Combining the two step clustering results, we discovered the six original clusters in the data. The result is shown in Fig. 3. This example demonstrates that if data contains a hierarchical structure of clusters, it is more effective to recover these clusters in multiple steps.

2.3 Interactive Approach

In contrast to the bottom-up hierarchical clustering methods, we use a top-down approach to interactively building cluster trees from data. Starting with the whole data set that is considered as a cluster on its own right, we stepwise decompose the data and grow a tree of clusters. In the tree, a node containing children is a composite cluster while all leaves are atomic clusters.

In this interactive approach, we have to make two decisions at each node to proceed the process. That is, to decide whether a node being considered is a composite or atomic cluster and to determine the number of clusters to be created from it if the node is a composite cluster. With synthetic data, since we know the details of clusters, we will have no difficulty to make these decisions. However, when dealing with real data, we usually have no knowledge about the structure of clusters The Fastmap algorithm and visualization help obtain such knowledge and make

decisions. Through Fastmap and other visualization methods, which we will discuss in Section 3, we can make decisions based on what we see. In clustering real world data, this kind of human involvement has a great advantage because we can use our domain knowledge to accept or reject the clusters generated by the clustering algorithm.

3 Cluster Validation with Fastmap

Cluster validation refers to the procedures that evaluate clusters created from a data set by a clustering algorithm [Jain88]. In this section, we briefly overview the Fastmap technique and discuss how to use it in cluster validation. We also present some other cluster validation methods.

3.1 Fastmap Algorithm

Given a set of N objects in an n dimensional space and their mutual distances, Fastmap projects the N objects onto a d ($< n$) dimensional space. The projection is performed in d steps. First, two objects O_a and O_b are selected as 'pivot objects' and a line considered passing through them forms the first axis in the projected d dimensional space. For any object O_i, its coordinate x_i on the first axis is calculated as

$$x_i = \frac{d_{a,i}^2 + d_{a,d}^2 - d_{b,i}^2}{2d_{a,b}} \qquad (1)$$

Here, $d_{a,b}$ is the distance between O_a and O_b, and $d_{a,i}$ and $d_{b,i}$ are the distances between O_i and O_a, O_b, respectively. Because all distances between objects are known, it is straightforward to calculate the coordinates of all objects on the first axis.

To calculate the coordinates of objects on the second axis, we first need to calculate the distances between objects in the reduced ($n - 1$) dimensional space. The distances can be calculated from the distances in the original n dimensional space and the coordinates on the first axis, as follows:

$$(d'_{i,j})^2 = (d_{i,j})^2 - (x_i - x_j)^2 \qquad i, j = 1, \ldots, N \qquad (2)$$

where $d'_{i,j}$ is the distance between obejcts O_i and O_j in the reduced ($n - 1$) dimensional space, $d_{i,j}$ is the distance between obejcts O_i and O_j in the original n dimensional space, x_i, x_j are the coordinates of O_i and O_j on the first axis. After the distances between objects in the reduced ($n - 1$) dimensional space are calculated, Equation (1) is used to calculate the coordinates on the second axis. This process is recursively used until the coordinates on the dth axis are calculated. For the details of the Fastmap algorithm, refer to [Folo95].

3.2 Cluster Validation with Fastmap

From the objects distribution viewpoint, validity of a cluster is measured by *Compactness* and *isolation* [Jain88]. Fastmap allows us to see the isolation of clusters generated from a high dimensional space. We use other methods to validate compactness of clusters, which will be discussed in Section 3.3.

To validate clusters generated by the *k*-prototypes algorithm from a high dimensional space, we use Fastmap to project objects in different clusters onto a 2D space and visualize them. In the 2D space, the cluster memberships of objects are maintained so we can display objects in different clusters with different symbols and colors. From the display, we can visually identify the clusters, which are isolated from others. For example, the display in Fig. 4 shows that clusters 2 and 4 are isolated from other clusters. We can validate that objects in clusters 2 and 4 form separate clusters in the original space.

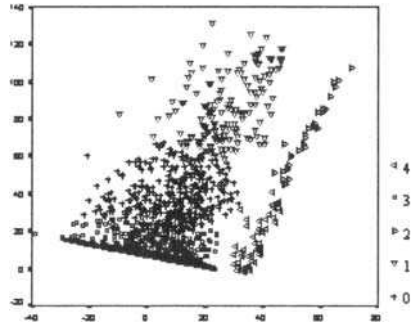


Fig. 4. The 2D display shows clusters 2 and 4 form separate clusters.

In Fig. 4, it is not clear whether clusters 0, 1 and 3 form clusters in the original space or not, because of the overlapping of objects. In this case, we can do the following:

- Remove clusters 2 and 4, and re-project others to see whether a clear separation can be observed.
- Remove clusters 2 and 4 and merge the rest. Apply the *k*-prototypes algorithm to partition the merge into another set of clusters. Then, use Fastmap to validate the new clusters.

Before clustering a data set, we can use Fastmap to project it onto a 2D space and see how many clusters possibly exist in the data set therefore we can determine the number of clusters to be generated.

3.3 Other Cluster Validation Methods

Fastmap helps identify isolated clusters generated from the original high dimensional space. However, it does not validate whether the isolated clusters are atomic or

composite. It cannot verify whether the overlapping clusters are separate in the original high dimensional space or not. In combination with Fastmap, we use the following validation methods to solve these two problems.

We use two methods to solve the first problem. Firstly, we visualize the characteristic information of isolated clusters. The characteristic information includes cluster centers, category distributions of categorical variables and histograms of numeric variables. The characteristic information can help understand the meanings of clusters. A cluster is found if it has a business meaning. Secondly, we visualize the histograms of distances of objects to cluster centers. From these histograms, we can visually evaluate the compactness of a cluster. For example, a cluster that has a small mean and standard deviation is more compact than the clusters with large means and standard deviations. In general, business meanings are most important in validating clusters.

We solve the second problem by visualizing the distances between clusters. We have developed a web view to visualize cluster relationships. An example is shown in Fig. 5. The filled circles represent clusters, which overlap on the 2D display. The thick lines linking clusters indicate two clusters are close. From this view, we can identify that cluster MR1980 and MR2098 are distant from others so we can use the above two methods to validate them independently. Other clusters can be merged into a composite cluster and further partitioned with the clustering algorithm.

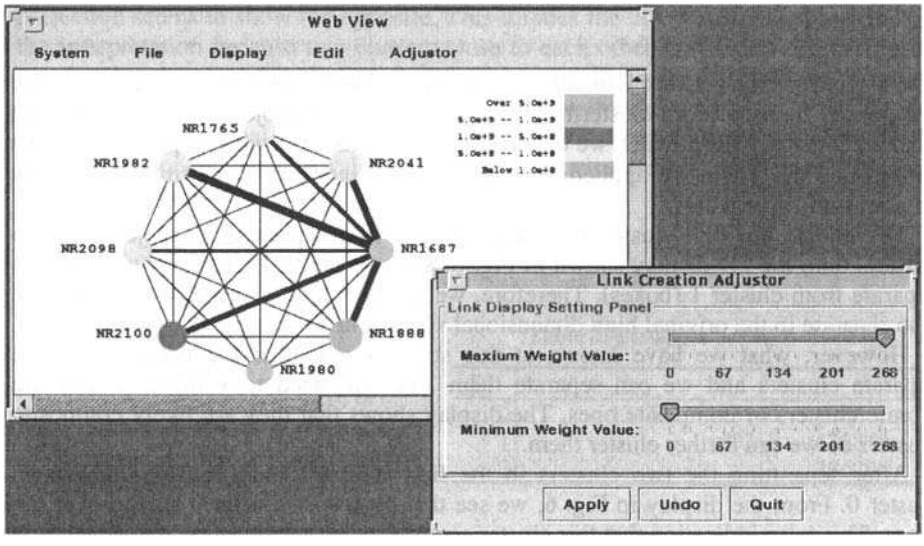


Fig. 5. A web view to visualize cluster relationships.

4 A Real Case Study

In this section, we use a real example to show how to use visual cluster validation methods to cluster real world data.

4.1 Business Problem and Data

In telecommunication service industry, customers often switch from one service provider to another. This phenomenon is referred to as churn. Churn costs providers dearly. Maintaining a low churn rate is a high priority in many companies.

Churned customers can be won back through winback marketing campaigns by offering them new benefits for reconnecting the service, such as low prices, free handsets, free minutes, etc. Because not every one prefers to the same offer, it is important to divide customers into similar groups and associate a suitable offer to each group in the campaign. Clustering provides an essential technique for customer segmentation.

The data set contains 1900 records, each representing a customer churned from a mobile service provider within two months. The customers are described in 37 fields, including customer ID, phone number, customer details such as age and sex, billing details such as payment methods and spending, service details as well as network usage details. After cleaning up the data, five fields were removed. Some are identification of customers and some contain only one value. In the remaining 32 fields, 28 fields are categorical and four fields are numeric. Fifteen fields contain missing values.

4.2 Cluster Analysis

The *k*-prototypes algorithm [Huan98] was used to cluster this data set because of the categorical fields. Before clustering, we used Fastmap to project the whole data set onto a 2D display from which we observed that two clusters could be produced in the first clustering. Then, we applied the *k*-prototypes algorithm to partition the whole data set into two clusters.

To validate our first clustering result, we used Fastmap again to project the two clusters onto the 2D display shown in Fig. 6. We can see that cluster 0 (triangles) is separate from cluster 1 (boxes). Therefore, we can conclude that the two clusters are also separate in the original high dimensional space. The clustering is justified.

However, what we have known at this stage is only that the two clusters are separate clusters and we can separate them. We still need to validate if they are atomic clusters or composite ones. The display shows that they are likely composite clusters so we can further cluster them.

After separating the two clusters in the first step, we only consider objects in cluster 0. From the display in Fig. 6, we see that objects in cluster 0 spread over the place. This is an indication that this cluster contains other sub clusters. Again, we first projected cluster 0 onto a 2D space to estimate the number of potential clusters, which happened to be 4. Then, we applied the *k*-prototypes algorithm to partition cluster 0 into four sub clusters. The projection of the four clusters on 2D is shown in Fig. 7. These clusters are separate in the display so they are also separate in the original space.

To validate these clusters, we further investigated their compactness and relationships. We calculated the means and standard deviations of distances of objects to the clusters centers (prototypes). The results are given in Table 1. These values indicate that sub cluster 0 is most compact because of its small standard deviation.

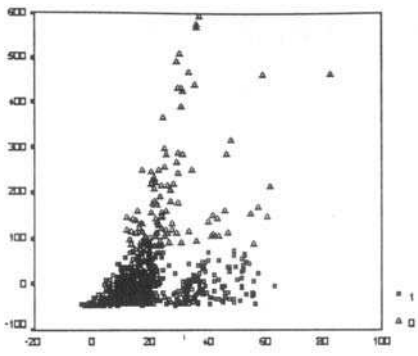


Fig. 6. The first clustering of the Winback data set.

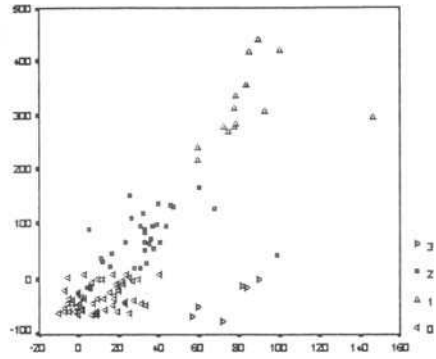


Fig. 7. Projection of four clusters created from cluster 0.

Sub cluster 1 is lest compact. Sub cluster 3 is more compact than sub cluster 2 although the later looks more compact than the former on the display.

To understand the relationships between these sub clusters, we calculated the distances between the centers of the clusters. The results are shown in Table 2. We can see that sub cluster 0 is closer to sub cluster 3 than to sub cluster 2 although the projection seems to show the opposite. This implies the 2D visualization can mislead the interpretation because two clusters close to each other on the 2D display do not warrant that they are also close in the original space. The only principle we can use is that if two clusters are separate in the projected low dimensional space, they are also separate in the original high dimensional space. When two clusters are close on the projected space, we need to validate them through visualization of their relationships in the original space.

Table 1. Cluster compactness.

Cluster	Mean	Std.
0	5527.0	1158.8
1	8319.3	5249.9
2	6303.3	2796.4
3	3156.2	1275.1

Table 2. Distances between clusters.

	0	*1*	*2*	*3*
0		131621	15140	7986
1			60368	128849
2				18620

By visualizing the characteristic information of these clusters, we found that all these four sub clusters were high spending clusters. However, the spending in sub clusters 0 and 3 was comparatively lower than the spending in sub clusters 1 and 2 in which sub cluster 1 was the highest. Interestingly, we found that although the spending in sub cluster 3 was low but its rate plan is higher than the rate plan of other clusters. Our result has also shown that all customers in sub cluster 3 were acquired from a particular sales channel. This may indicate that the sales channel didn't help customers choose the right service if they were not cheating customers. If true, actions need to be taken to correct their business practice.

After we finished the analysis of cluster 0 in the first level of clustering, we started to use the same methods to analyze cluster 1. Before clustering, we used Fastmap to estimate the number of potential clusters. Then, we applied the *k*-prototypes algorithm to partition cluster 1 into five clusters. Using Fastmap and other validation methods, we found that two of them form atomic clusters and the other three contain sub clusters, which need to be further discovered. We continued this process until we finally got 19 atomic clusters from the whole data set.

Fig. 8 shows the cluster tree discovered using the interactive approach. In this tree diagram, each box represents a node, which is a cluster of customers. The number inside each box is the number of customers in that cluster. The R# index uniquely identifies each cluster in the process. The gray nodes represent composite clusters that contain sub clusters. The white nodes indicate atomic clusters. The root of the tree R0 represents the entire data set.

After the tree was constructed, we can easily validate any single cluster or group of the clusters selected from the tree. We can visualize the characteristic information of the clusters to understand the meanings of the clusters and identify important and interesting customer segments to be targeted in the winback campaigns.

We have found that the Fastmap 2D projection often gives good indications to interesting clusters. For example, from the Fastmap projection, cluster R27 has a special distribution (plus signs in Fig. 9). After analyzing their characteristic information, it turned out that this cluster contains customers who are instant churners. The characteristics of these people are that they joined the service, spent little and then left after a few months. Due to the size of this cluster, special campaigns were designed to target these customers.

5 Conclusions

In this paper, we have presented a visual method of cluster validation using Fastmap. Instead of using it to produce low dimensional data for clustering, we use Fastmap to validate clusters created by a clustering algorithm from the original high dimensional space. The principle here is that if clusters are separate in the projected low dimensional space, they are also separate in the original space but the opposite is not true. This enables us to identify the separate clusters in the original high dimensional space and focus on the overlapping clusters. Fastmap also enables us to determine the number of potential clusters to produce before the clustering algorithm is run.

The interactive approach to building cluster trees and discover interesting clusters with various visual cluster validation methods has proved effective in clustering large data sets in real world databases. We have presented a real case study on how to use this approach to cluster complex real data. In the case study, we have demonstrated that Fastmap and other cluster validation methods can be used effectively to verify clustering results and validate clusters produced by the *k*-prototypes clustering algorithm at different steps. Our results have shown that their use in the interactive approach is necessary when dealing with complex data because useful clusters are difficult to discover from a single run of clustering. We have developed a prototype system to facilitate these analysis activities.

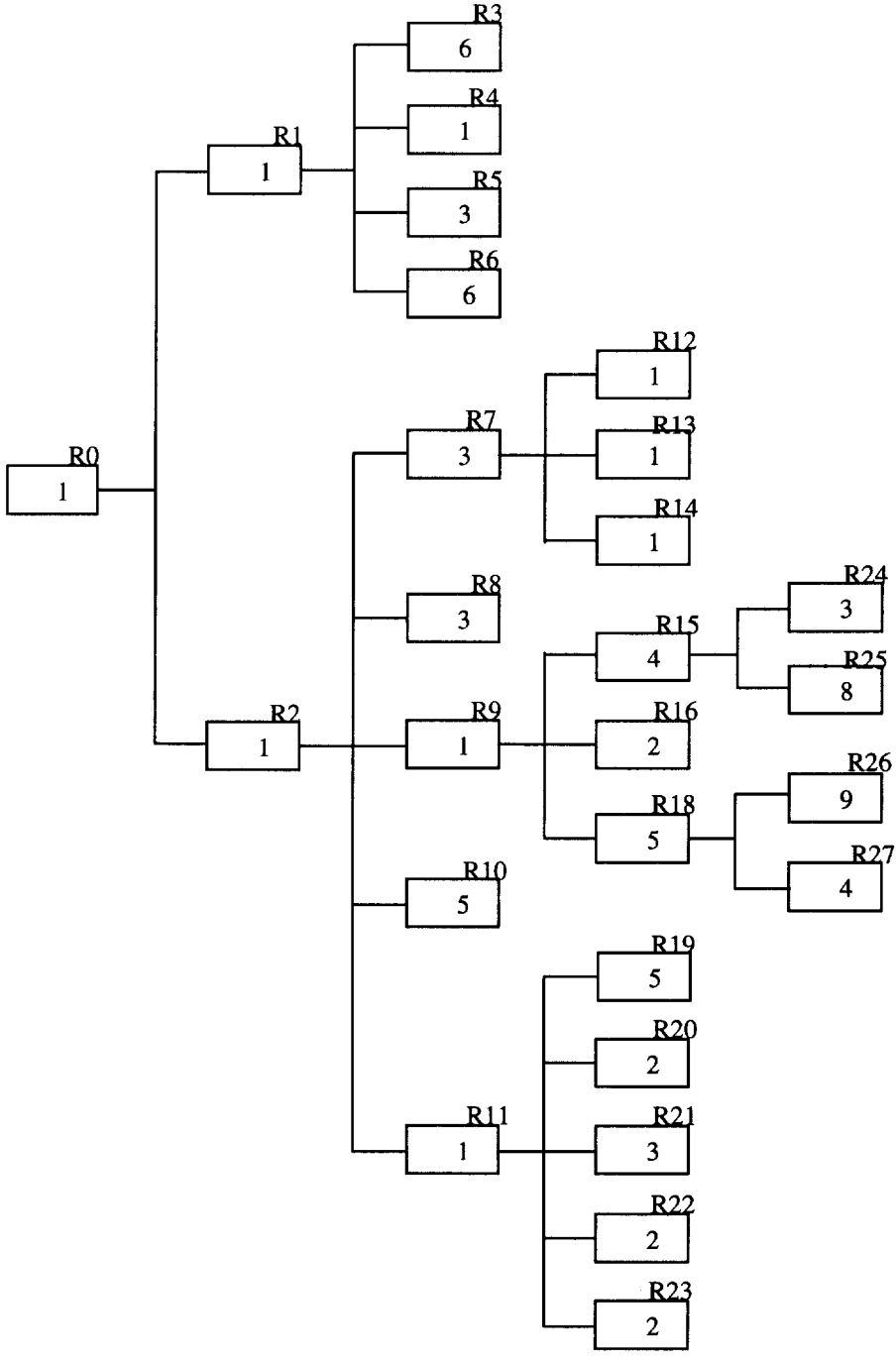


Fig. 8. The cluster tree interactively created from the Winback data set. The number inside each box is the number of customers in that cluster. R# index uniquely identifies each cluster.

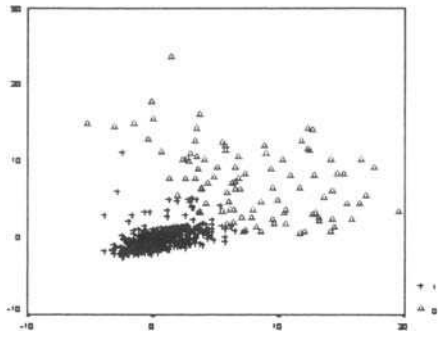


Fig. 9. Projection of clusters R26 (0) and R27 (1).

References

[Agra98] Agrawal, R., Gehrke, J, Gunopulos, D. and Raghavan, P. (1998) Automatic subspace clustering of high dimensional data for data mining applications. In Proceedings of SIGMOD Conference.

[Este96] Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, Portland, Oregon, USA.

[Falo95] Faloutsos, C. and Lin, K., (1995) Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Proceedings of ACM-SIGMOD, pp. 163-174.

[Fuku90] Fukunaga, K. (1990) *Introduction to Statistical Pattern Recognition.* Academic Press.

[Gant99] Ganti, V., Ramakrishnan, R., Gehrke, J, Powell, A. L. and French, J. C. (1999) Clustering large datasets in arbitrary metric spaces. ICDE 1999, pp. 502-511.

[Huan98] Huang, Z. (1998) Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, Vol. 2, No. 3, pp. 283-304.

[Jain88] Jain, A. K. and Dubes, R. C. (1988) *Algorithms for Clustering Data.* Prentice Hall.

[NgHa94] Ng, R. and Han, J. (1994) Efficient and effective clustering methods for spatial data mining. In Proceedings of VLDB, 1994.

[Theo99] Theodoridis, S. and Koutroumbas, K. (1999) *Pattern Recognition.* Academic Press.

[Youn87] Young, F. W. (1987) *Multidimensional Scaling: History, Theory and Applications.* Lawrence Erlbaum Associates.

[Zhan97] Zhang, T. and Ramakrishnan, R. (1997) BIRCH: A new data clustering algorithm and its applications. Data Mining and Knowledge Discovery, Vol. 1, No. 2, pp. 141-182.

COE: Clustering with Obstacles Entities A Preliminary Study

Anthony K. H. Tung, Jean Hou, and Jiawei Han

School of Computing Science
Simon Fraser University
British Columbia
Canada V5A 1S6
{khtung,jhou,han}@cs.sfu.ca

Abstract. Clustering analysis has been a very active area of research in the data mining community. However, most algorithms have ignored the fact that physical obstacles exist in the real world and could thus affect the result of clustering dramatically. In this paper, we will look at the problem of clustering in the presence of obstacles. We called this problem the COE (Clustering with Obstacles Entities) problem and provide an outline of an algorithm called COE-CLARANS to solve it.

1 Introduction

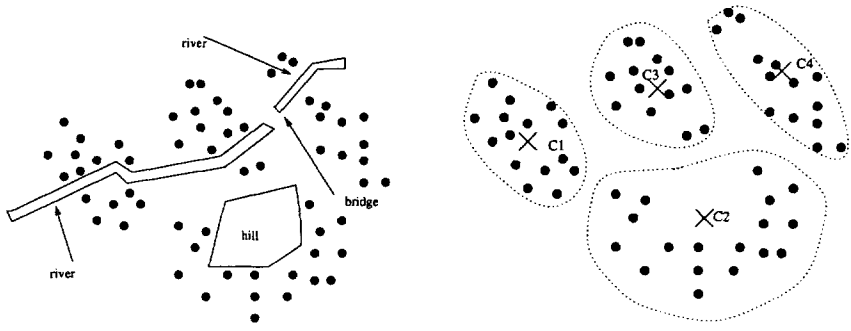
The studies of clustering on large databases started with the introduction of CLARANS [NH94] and since then, a tremendous amount of research had been made by the database community on this field [HK00].

Typically, a clustering task consists of separating a set of points into different groups according to some *measure of goodness* that differ according to application. For example, in market research, managers who are planning the location of their stores may wish to cluster their customers according to their location and then locate a store to serve each cluster. In such a case, a common measure of goodness will be the sum of square of the **direct** Euclidean distance between the customers and the centre of the cluster they belong to. However, in many real applications, the use of direct Euclidean distance has its weakness as illustrated by the following example.

Example 1. A bank manager wishes to locate 4 ATMs in the area shown in Figure 1a to serve the customers who are represented by points in the figure. In such a situation, however, natural obstacles exist in the area and they should not be ignored. This is because ignoring these obstacles will result in clusters like those in Figure 1b which are obviously wrong. Cluster C_1, for example, is in fact split by a river and some customers on one side of the river will have to travel a long way to the allocated ATM on the other side of the river.

Example 1.1 illustrated a practical problem encountered by many users of traditional clustering algorithms: the lack of mechanism to integrate physical

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 165–168, 2000.
© Springer-Verlag Berlin Heidelberg 2000



(a) Customers' location and obstacles.

(b) Clusters formed when ignoring obstacles.

Fig. 1. Planning the location of ATMs

obstacles into clustering algorithms. In many application, the discovered clusters can be much more useful if they are found while keeping the physical imitation of the obstacles in mind.

Depending on the application on hand, different clustering algorithms will be needed and they will be affected differently by the existence of obstacle entities. In this paper, we will concentrate on adapting CLARANS to handle obstacles and we called the adapted algorithm **COE-CLARANS**. The problem in Example 1.1 is formally described as follows:

We are given a set P of n points $\{p_1, p_2, ..., p_n\}$ and a set O of m **non-intersecting** obstacles $\{o_1, ..., o_m\}$ in a two dimensional region, R with each obstacle o_i represented by a simple polygon. The distance, $d(p, q)$ between any two points, p and q, is defined as the length of the shortest Euclidean path from p to q without cutting through any obstacles. To distinguish this distance from the direct Euclidean distance, we will refer to this distance as *obstructed distance* in this paper. Our objective is to partition P into k clusters $C_1, ..., C_k$ such that the following square-error function, E, is minimized.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} d^2(p, m_i)$$

where m_i is the centre of cluster C_i that is determined also by the clustering.

Due to lack of space, we will only outline the steps taken in COE-CLARANS to handle obstacles in the next section follow by the conclusion in Section 3.

2 The COE-CLARANS Algorithm

In order to adapt an existing clustering algorithm like CLARANS to handle obstacles, two different approaches can be adopted. The first is a loosely-coupled

approach in which the obstacles are handled solely by the distance function and the clustering algorithm uses the distance function as a black box without catering for obstacles. The second approach is a tightly-coupled approach in which both the clustering algorithm and the distance function take obstacles into account. COE-CLARANS uses the second approach as it give more room for optimizing performance. COE-CLARANS use two techniques to perform efficient clustering. We will introduce them in this section.

2.1 Pre-clustering

To make COE-CLARANS efficient, a pre-clustering step similar to those in BIRCH [ZRL96], ScaleKM [BFR98] and CHAMELEON [KHK99] are taken to group the objects into a set of *clustering features* [ZRL96]. We call these clustering features, *micro-clusters*. There are two advantages in adding a pre-clustering step. First, the compressed micro-clusters take up much less memory space and clustering can thus be performed in main memory. Second, as computing the distance between objects and the cluster centers is an expensive operation, pre-clustering will help reduce the number of such operation.

In order to avoid having micro-clusters that are split by an obstacle, we first triangulate the region R into triangles and group the data points according to the triangle that they are in. Micro-clusters are then formed in each group separately. As points within a triangle are all mutually visible to each other, this ensures that micro-cluster formed are not split by an obstacle.

With the use of micro-clusters for clustering, we have to take note that the cluster centers are now micro-clusters and we are approximating the location of the actual medoids to be within these cluster centers.

2.2 Using the Lower Bound of E for Pruning

The CLARANS algorithm is a generate-and-test algorithm which randomly pick a cluster center o_i and try to replace it with a new center o_{random}. To judge whether o_{random} is a better center than o_i, the square error function E is computed with o_{random} as the cluster center and if it is found to be lower than the one computed with o_i as the center, replacement will take place. However, the computation of E is very expensive with the existence of obstacles. To avoid the unnecessary computation of E, an more easily computed **lower bound** of E, E' is first computed. If E' is already higher than the best solution so far, then o_{random} can be abandoned without the need for E to be computed.

To compute E' with o_{random} as a cluster center, we first underestimate the distance between o_{random} and the micro-clusters by using direct Euclidean distance. Thus, if the direct Euclidean distance between a micro-cluster p and o_{random} is shorter than the obstructed distance between p and the other $k-1$ unchanged cluster centers, then p is assigned to o_{random} and the direct Euclidean distance between them will be used when computing the estimated square-error

function E'. This makes E' a lower bound for the actual square-error function E. Since E' is a lower bound of E, we can choose to abandon o_{random} without computing E if E' is already higher than the square-error function of the best solution so far.

3 Conclusion

In this paper, we introduce the problem of COE which we believe is a very real and practical problem. We selected a clustering problem and outline an algorithm COE-CLARANS for solving it. COE-CLARANS makes use of two main ideas to enhance its efficiency. First, it uses the idea of pre-clustering to compress the dataset into micro-clusters which could be clustered in the main memory and thus avoids I/O overhead. Second, it avoids unnecessary computation by first estimating a lower bound E' for the square-error function E and then computes E only if E' proves to be lower than the best solution that has been found. We believe that there is still a lot of room for research in the problem of COE and hope that our work could motivate more people to look into this area.

References

[BFR98] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 9–15, New York, NY, August 1998.

[HK00] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* (to be published by) Morgan Kaufmann, 2000.

[KHK99] G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. *COMPUTER*, 32:68–75, 1999.

[NH94] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 144–155, Santiago, Chile, September 1994.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, pages 103–114, Montreal, Canada, June 1996.

Combining Sampling Technique with DBSCAN Algorithm for Clustering Large Spatial Databases[1]

Shuigeng Zhou, Aoying Zhou, Jing Cao, Jin Wen, Ye Fan, and Yunfa Hu

Computer Science Department, Fudan University, Shanghai, 200433, China
970218@fudan.edu.cn

Abstract. In this paper, we combine sampling technique with DBSCAN algorithm to cluster large spatial databases, two sampling-based DBSCAN (SDBSCAN) algorithms are developed. One algorithm introduces sampling technique inside DBSCAN; and the other uses sampling procedure outside DBSCAN. Experimental results demonstrate that our algorithms are effective and efficient in clustering large-scale spatial databases.

1 Introduction

DBSCAN [1] is a high-performance clustering algorithm that can discover clusters of arbitrary shape and handle the noise points effectively. However, for large-scale spatial databases, DBSCAN requires large volume of memory support and could incur substantial I/O costs because it operates directly on the entire database. The aim of this paper is to extend the DBSCAN algorithm to cluster large-scale spatial databases by data sampling technique. Two novel sampling-based clustering algorithms are proposed and implemented by combining the sampling technique with DBSCAN. One algorithm introduces sampling technique inside DBSCAN and the other applies sampling procedure outside DBSCAN. Owing to data sampling, the I/O cost and memory requirement for clustering are reduced dramatically, and the run-time of clustering is thus cut down considerably. Experimental results demonstrate that our approach is effective and efficient in clustering large-scale spatial databases.

2 Sampling-Based DBSCAN Algorithms

While handling large-scale databases or data warehouses, one common used technique in clustering analyses is data sampling, which selects a relatively small number of representatives from databases or data warehouses and applies the clustering algorithms only to these representatives. However, to the best of our knowledge, no research on combining sampling technique with DBSCAN has been reported. We develop two sampling-based DBSCAN (SDBSCAN) algorithms. One

[1] This work was supported by the NSF of China (Grant no. 69743001).

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 169-172, 2000.
© Springer-Verlag Berlin Heidelberg 2000

SDBSCAN algorithm adopts sampling technique inside DBSCAN, i.e. *inside sampling* approach, and the other SDBSCAN algorithm uses sampling procedure outside DBSCAN, i.e. *outside sampling* approach. Comparing with other clustering algorithms using sampling technique, our approaches have two outstanding features:

- Sampling technique and clustering algorithm are bounded together.
- Clustering not only over the sampled data set, but also over the whole data set.

For more details about SDBSCAN algorithms, the readers can refer to [2].

2.1 The Idea of Sampling Inside DBSCAN Algorithm: SDBSCAN-1

DBSCAN selects a global k-dist value for clustering. For the thinnest clusters, the number of objects contained in their core objects' neighborhoods with radius *Eps* equal to k-dist is *k*. However, for the other clusters, the number of objects contained in most of their core objects' neighborhoods of the same radius is more than *k*. DBSCAN carries out *region query* operation for every object contained in the core object's neighborhood. For a given core object *p* in cluster *C*, it's conceivable that the neighborhoods of the objects contained in *p* will intersect with each other. Suppose *q* is an object in *p*'s neighborhood, if its neighborhood is covered by the neighborhoods of other objects in *p*, then the *region query* operation for *q* can be omitted because all objects in *q*'s neighborhood can be fetched by the *region queries* of the other objects in *p*, which means that *q* is not necessary to be selected as a seed for cluster expansion. Therefore, both time-consuming on *region query* operation for *q* and memory requirement for storing *q* as a core object can be cut down. In fact, for the dense clusters, quite a lot of objects in a core object's neighborhood can be ignored being chosen as seeds. So for the sake of reducing memory usage and I/O costs to speed up the DBSCAN algorithm, we should sample some representatives rather than take all of the objects in a core object's neighborhood as new seeds. We call these sampled seeds *representative object* of the neighborhood where these objects are held. Intuitively, the outer objects in the neighborhood of a core object are favorable candidates of *representative object* because the neighborhoods of inner objects tend to being covered by the neighborhoods of outer objects. Hence, sampling the representative seeds is in fact a problem of selecting representative objects that can accurately outline the neighborhood shape of a core object.

2.2 The Idea of Outside Sampling DBSCAN Algorithm: SDBSCAN-2

Outside sampling is in fact a traditional technique. However, in our SDBSCAN-2 algorithm, a novel and efficient labeling mechanism is adopted to implementing the labeling process of the un-sampled data based on R*-tree. The scheme of SDBSCAN-2 is like this:

- Sample database *DB* to produce sampled dataset *sdb*,
- Create R*-trees for *DB* and sampled data set *sdb*,
- Cluster sampled data set *sdb* with DBSCAN,
- FOR each core point *p* in sampled data set *sdb* DO:
  - resultP:=*DB*.regionquery (*p*, *Eps*),
  - *DB*.changeClIds (resultP, *p*.ClId).

Step 1 is the sampling procedure. A sampling algorithm for drawing a sample randomly from data in file in one pass and using constant space is used. In order to guarantee the clustering quality, an analytical limitation on minimum sampled data amount is applied. Step 2 is responsible for building R*-trees for *DB* and the sampled data set *sdb*. Step3 and 4 are the key steps that are used for clustering and labeling respectively. We cluster the sampled data set *sdb* with DBSCAN algorithm. Once a core point is found in *sdb*, all points in its neighborhood of the same radius in *DB*, UNCLASSIFIED or CLASSIFIED as NOISE, sampled or un-sampled, are labeled as members of the current cluster. Therefore, the clustering process (over the sampled data set *sdb*) and the labeling process (over un-sampled points in *DB*) are in fact carried out concurrently. When clustering is over, labeling is also finished. A further improvement on SDBSCAN-2 algorithm is as follows. While building R*-tree, we don't create a separate R*-tree for the sampled data set *sdb*. Instead, we build only one R*-tree for *DB*. In other words, we merge the R*-trees of *DB* and *sdb* in the former version of SDBSCAN-2 into one single R*-tree, in which we mark out which point is sampled and which is not. Therefore, the operations of clustering and labeling are carried out over the same R*-tree. And for each core point in *sdb* only one time of region query operation is executed, unlike in the former version of SDBSCAN-2 where two times is needed: the first time for clustering over the R*-tree of *sdb*, and the second time for labeling over the R*-tree of *DB*. Obviously, this improvement can cut down almost half of the region queries.

3 Performance Evaluation

We evaluate SDBSCAN algorithms with both synthetic sample databases and the database of the SEQUOIA 2000 benchmark. Generally, SBSCAN algorithms can be faster than DBSCAN by several times. Figure 1 illustrates scale-up experiment with DBSCAN, SDBSCAN-1 and SDBSCAN-2, which shows that SDBSCAN algorithms have better scalability over data set size than DBSCAN. Here, sampling ratio is 20%.

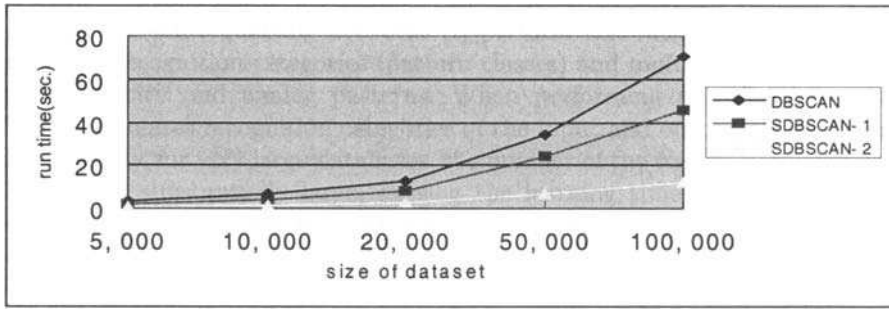


Fig. 1. Performance comparison: DBSCAN, SDBSCAN-1 and SDBSCAN-2

References

1. M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of 2nd International Conference on Knowledge Discovering in Databases and Data Mining (KDD-96), Portland, Oregon, August 1996.
2. S. Zhou, *et al.* Combining sampling technique with DBSCAN algorithm for clustering large spatial databases. Technical Report of Computer Science Department, Fudan University, 1999.

Predictive Adaptive Resonance Theory and Knowledge Discovery in Databases

Ah-Hwee Tan[1] and Hui-Shin Vivien Soon[2]

[1] Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613
ahhwee@krdl.org.sg
[2] Ngee Ann Polytechnic, 535 Clementi Road, Singapore 599489
shs@np.edu.sg

Abstract. This paper investigates the scalability of predictive Adaptive Resonance Theory (ART) networks for knowledge discovery in very large databases. Although predictive ART performs fast and incremental learning, the number of recognition categories or rules that it creates during learning may become substantially large and cause the learning speed to slow down. To tackle this problem, we introduce an on-line algorithm for evaluating and pruning categories during learning. Benchmark experiments on a large scale data set show that on-line pruning has been effective in reducing the number of the recognition categories and the time for convergence. Interestingly, the pruned networks also produce better predictive performance.

1 Introduction

One of the major challenges faced by the predictive modeling techniques is the efficiency and the scalability to very large databases. Gradient descent based neural network models require many learning iterations through the training data and are highly computational intensive. This paper presents an alternative approach to knowledge discovery using a predictive self-organizing neural network model, known as the Adaptive Resonance Associative Map (ARAM) [5].

Predictive self-organizing networks [1] perform fast incremental supervised learning of recognition categories (pattern classes) and multi-dimensional mappings of binary and analog patterns. When performing classification tasks, ARAM formulates recognition categories of the input and output pattern pairs. Unfortunately, for very large databases, the number of the recognition categories may become substantially large, causing the learning time to increase significantly. To tackle this problem, we introduce an on-line algorithm to estimate the *merit* or *confidence values* of the recognition categories of an ARAM network during learning. Each newly created category node is given a confidence value of 1. A *forgetting* process constantly reduces confidence values towards 0 at a specific interval. In conjunction, a *reinforcement* process increases confidence values towards 1s when correct predictions are produced by their respective recognition categories. The confidence values of the category nodes are then compared with

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 173–176, 2000.
© Springer-Verlag Berlin Heidelberg 2000

a threshold parameter. Categories with confidence values below the threshold are removed from the network.

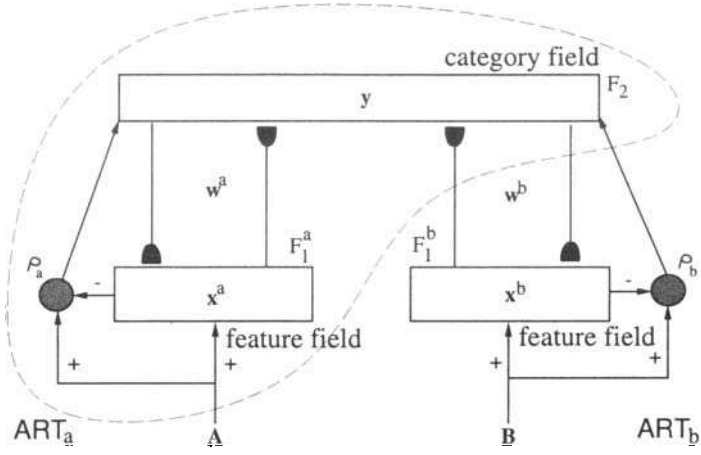


Fig. 1. The Adaptive Resonance Associative Map architecture.

2 Fuzzy ARAM

An ARAM network (Figure 1) consists of two input feature fields, F_1^a and F_1^b, connected by bi-directional and conditionable pathways to a category field F_2. For classification problems, F_1^a serves as the input field representing input activity vectors and F_1^b serves as the output field representing output class vectors.

Given a pair of input and output patterns, ARAM computes a choice function for each F_2 recognition category. The winning node that has the maximal choice function value then triggers a top-down priming on F_1^a and F_1^b to check if its associated input and output weight vectors satisfy the vigilance criteria in their respective modules. If so, under fast learning, the weight templates of the F_2 recognition category are modified towards their intersection with the input and output vector pair. Otherwise, the recognition category is reset and the system repeats to select another category node until a match is found. By synchronizing the unsupervised clustering of the input and output pattern sets, ARAM learns supervised mapping between the input and output patterns. As code stabilization is ensured by restricting learning to states where resonances are reached, fast learning in a real-time environment is feasible. Please refer to [5] for the detailed algorithm.

3 ARAM Complexity and Category Pruning

Let P be the number of the input and output training pattern pairs, Q be the number of the recognition categories created by an ARAM network, M be the

number of the input attributes, and N be the number of the output classes. The time complexity per learning iteration of the ARAM algorithm is given by $O(PQ(M+N))$. Since M and N are typically fixed for a specific knowledge discovery task, the complexity thus depends on P and Q. When Q is small, say around $\log(P)$, the time complexity is $O(P\log P)$. However, if Q is large, the complexity grows to P^2 in the worst case.

To improve learning efficiency, we propose a method for evaluating and eliminating categories that are created by spurious cases during learning. Each category node j is associated with a confidence factor c_j, a real number between 0 and 1. For a newly committed node j, c_j equals 1. At a fixed interval, a *forgetting* process constantly causes c_j to decay towards 0. A *reinforcement* process increases c_j towards 1 whenever a correct prediction is made by the category node j during learning.

Confidence erosion: At a chosen interval, the confidence value of each recognition category depreciates towards 0 according to

$$c_J^{(\text{new})} = c_J^{(\text{old})} - \zeta c_J^{(\text{old})}, \tag{1}$$

where $\zeta \in [0,1]$ is the confidence decay parameter. The erosion process is self-scaling in the sense that the decay becomes smaller as c_J gets smaller.

Confidence reinforcement: When a category node J, chosen by the code competition process, makes a correct prediction, its confidence value c_J is increased towards 1 by

$$c_J^{(\text{new})} = c_J^{(\text{old})} + \eta(1 - c_J^{(\text{old})}), \tag{2}$$

where $\eta \in [0,1]$ is the confidence reinforcement parameter.

Category pruning: The computed confidence values are then compared with a threshold parameter $\tau \in [0,1]$. A category is removed from the ARAM network when its confidence value falls below τ.

Convergence criterion: After each training iteration, the pruned network is evaluated against the training set for its predictive performance. Training is stopped when the improvement on the training set performance (in terms of percents, compared with that obtained in the previous iteration) falls below a convergence threshold ξ.

4 Experiments

The adult data set [4] is one of the largest public domain data set. It contains 48,842 records, each characterized by six continuous attributes and eight nominal features. The task is to predict whether a person with a particular set of personal attributes draws a salary greater or less than US$50,000. The adult data set was noted as a hard domain with a good number of records and a mix of continuous and discrete features.

Fuzzy ARAM experiments used a standard set of parameter values: choice parameters $\alpha_a = \alpha_b = 0.1$, learning rates $\beta_a = \beta_b = 1.0$, and contribution parameter $\gamma = 1.0$. For on-line pruning, we used $\zeta = 0.005$ for confidence decay,

$\eta = 0.5$ for confidence reinforcement, $\tau = 0.05$ for pruning threshold, and $\xi = 0.5$ for stopping criterion. For prediction, ARAM used K-max rule [5] with $K = 3$.

Table 1. *Simulation results of fuzzy ARAM on the adult data set compared with those of KNN, C4.5, and NBTree. A '-' indicates that a value is not available for comparison. The results of ARAM were averaged over 10 simulations.*

Methods	# Epochs	# Nodes/ Rules	Time (Secs)	Test Accuracy
1-Nearest-Neighbor	1	30162	-	78.6
3-Nearest-Neighbor	1	30162	-	79.7
C4.5	-	2213	-	84.6
NBTree	-	137	-	85.9
Fuzzy ARAM	11.5	3826	8226	81.0
+ On-line pruning	7.5	343	1125	84.1

As shown in Table 1, fuzzy ARAM created a total of 3,826 category nodes from the 30,152 training patterns, with a compression ratio of around 8. Each learning process took an average of 11.5 iterations to converge. One complete benchmark experiment, including training and testing, took 8373 seconds (more than two hours) using an Ultra-60 SUN SPARC machine. With on-line category pruning, fuzzy ARAM produced an average of 343 categories. The networks also converged faster in an average of 7.5 iterations. One complete benchmark involving 30,152 training cases and 15,060 test cases took about 1,125 seconds (less than 20 minutes). Comparing predictive performance, fuzzy ARAM obtained an accuracy of 81.0% on the test cases, about the same as those of K-Nearest-Neighbor. With on-line category pruning, the test set accuracy improved to 84.1%, roughly comparable to those obtained by C4.5 and NBTree [3].

References

1. Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., Rosen, D. B.: Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, **3** (1992) 698–713
2. Carpenter, G. A., Tan, A.-H.: Rule extraction: From neural architecture to symbolic representation. *Connection Science*, **7** (1995) 3–27
3. Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. *Proceedings, KDD-96* (1996)
4. Murphy, P. M., Aha, D. W.: UCI repository of machine learning databases [machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science (1992)
5. Tan, A.-H.: Adaptive Resonance Associative Map. *Neural Networks* **8** (1995) 437–446

Improving Generalization Ability of Self-Generating Neural Networks Through Ensemble Averaging

Hirotaka Inoue and Hiroyuki Narihisa

Department of Information & Computer Engineering
Okayama University of Science,
1-1 Ridai-cho, Okayama, 700-0005, Japan
{inoue,narihisa}@ice.ous.ac.jp

Abstract. We present an ensemble averaging effect for improving the generalization capability of self-generating neural networks applied to classification problems. The results of our computational experiments show that ensemble averaging effect is 1-7% improvements in accuracy comparing with single SGNN for three benchmark problems.
Keywords: self-generating neural networks, self-generating neural tree, ensemble averaging, classification, competitive learning

1 Introduction

Self-generating neural networks (SGNNs) are focussed an attention because of their simplicity on networks design [1]. SGNNs are some kinds of extension of the self-organizing maps (SOMs) of Kohonen [2] and utilize the competitive learning algorithm which is implemented as self-generating neural tree (SGNT).

The SGNT algorithm is proposed in [3] to generate a neural tree automatically from training data directly. In our previous study concerning the performance analysis of the SGNT algorithm [4], we showed that the main characteristic of this SGNT algorithm was its high speed convergence in computation time but it was always not best algorithm in its accuracy comparing with the existing other feed-forward neural networks such as the backpropagation (BP) [5].

In this paper, we present the effect on ensemble averaging for improving the accuracy of self-generating neural networks (SGNNs) on classification problems. In order to investigate the effect of the ensemble averaging, we compare this model with the single SGNN model. In our study, we apply to standard classification problems MONK's [6], Cancer [7], Card [7], which are given as benchmarks.

2 Ensemble Averaging of SGNNs

SGNNs are directly learned not only the weights of the network connections but also the structure of the whole network from the given input data. In order to

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 177–180, 2000.
© Springer-Verlag Berlin Heidelberg 2000

decide a winner neuron n_{win} for training p dimensional data $\boldsymbol{e}_i$, the competitive learning is used. If a neuron n_j in the current SGNT includes the n_{win} as its descendant, weights w_{jk} of the n_j are updated as follows:

$$w_{jk} \leftarrow w_{jk} + \frac{1}{c_j + 1} \cdot (e_{ik} - w_{jk}). \tag{1}$$

here, k is from 1 to p, and c_j is the number of the leaf neurons in n_j.

After all input data are inserted into the SGNT as its leaf neurons, the weights of each node neuron n_j are the averages of the corresponding weights of all its children. Whole SGNT reflect the given feature space by its topology.

Though SGNNs have an ability of fast learning and an applicability of large scale problems, the accuracy of the classification is not so good as feed-forward networks which are implemented as a supervised learning method like the BP. In order to acquire more higher performance from given training data, we consider an ensemble averaging of K SGNTs.

The structure of the SGNT changes dynamically in training. The SGNT algorithm decide the structure of the SGNT after all training data are added in the SGNT. The different structure of the SGNT is generated by changing the input order of the training data.

In training process, we define "shuffler" to shuffle the set of input data E. The set of all input training data E enters each SGNN through each shuffler. The shuffler makes shuffle elements of E at random. All SGNTs are generated by adopting the SGNT algorithm. After training process, various SGNTs are generated independently.

In testing process, the set of test data T entered this ensemble model. Each output vector $\boldsymbol{o}_k \in \Re^N$ denotes the output of the expert k for the set of test data T. The output of this ensemble model is computed by averaging the each expert output as follows:

$$\boldsymbol{o} = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{o}_k. \tag{2}$$

Additionally, each expert can train and test independently. This model has a possibility of a parallel computation at the training process and the testing one.

3 Experimental Results

In order to analyze the generalization capability of SGNNs through ensemble averaging, we select three typical classification problems which are given as benchmark problems in this classification field. Table 1 shows the dataset summary. All problems are binary classification problems. We evaluate the classification accuracy by comparison with ensemble SGNNs and the single SGNN. The classification accuracy of each network is the percentage of correctly recognized examples in the set of all examples.

On the SGNT algorithm, the number of SGNTs K for ensemble averaging is changed from 1 to 100 (1,2,3,4,5,6,7,8,9,10,15,20,25,30,35,40,45,50,75, and

Table 1. Dataset Summary. #Data - data size (training data size, test data size), Type - attribute type, and #A - number of attributes.

Name		#Data	#A	Type
Monk's	M_1	432 (124,432)	6	discrete
	M_2	432 (169,432)	6	discrete
	M_3	432 (122,432)*	6	discrete
Cancer		699 (350,174)	9	continuous
Card		690 (345,172)*	51	mixed

100). Our experimental results are computed by IBM PC/AT (CPU: Pentium II 450MHz, Memory: 192MB). We compute 50 trials for each single/ensemble method.

Fig. 1 (a), (b), and (c) shows the relation between the number of SGNTs and the classification accuracy of the ensemble SGNNs for MONK's, Cancer, and Card respectively. Here, each classification accuracy is the average of 50 trials. In Fig. 1, it is shown that the classification accuracy are improved by computing the ensemble averaging of various SGNTs for all problems. The improvement ability is obtained from 2 to 10 SGTNs most effectively. The classification accuracy of larger than 20 SGNTs ensemble model is convergence for all problems. The improvement ability is obtained from 2 to 10 SGTNs most effectively. The classification accuracy of Larger than 20 SGNTs ensemble model is convergence for all problems. From these results, it seems to be decided the number of SGNTs K is about from 10 to 20 concerning the improvement rate and computation time.

Table 2 shows comparisons of the classification accuracy and the computation time of the single SGNN and best classification accuracy of the ensemble model. All results are showed the average of each trial. In Table 2, the ensemble model of SGNNs is better classification accuracy than the single SGNN at all problems in ave. Concerning M_1, M_2, and M_3, improving of the classification accuracy are about 7.2%, 2.4%, 4.7% respectively. Concerning Cancer, although all classification results show good results about 95% for the single SGNN, ensemble models improve the classification accuracy 0.9-2.2%. Concerning Card problem, the improvement of the classification accuracy are about 1.2-2.5%.

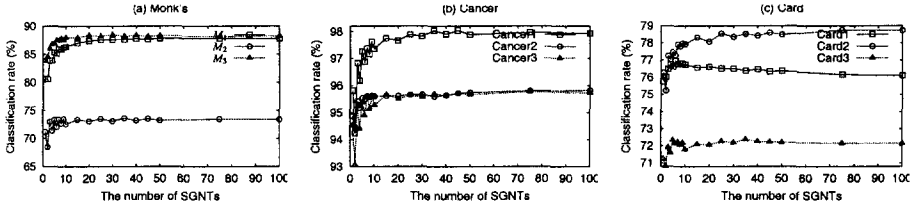


Fig. 1. Relation between the number of SGNTs and the classification accuracy of the ensemble SGNNs

Table 2. Comparisons of the classification accuracy and the computation time (in sec.) of the single SGNT and the best classification accuracy of the ensemble model

	M_1	M_2	M_3	Cancer1	Cancer2	Cancer3	Card1	Card2	Card3
single	80.6%	71.2%	83.8%	95.8%	94.9%	94.5%	75.7%	76.3%	71.2%
time(sec.)	0.06	0.08	0.06	0.12	0.12	0.12	0.46	0.44	0.44
ensemble	87.8%	73.6%	88.5%	98.0%	95.8%	95.7%	77.0%	78.8%	72.4%
time(sec.)	3.13	2.73	1.95	4.25	12.56	8.92	2.78	33.10	15.58

4 Conclusions

In this paper, we presented the effect on ensemble averaging of SGNNs to improve the classification accuracy. Experimental results show that ensemble averaging of SGNNs improve 1-7% increase of classification accuracy comparing with the single SGNN by sacrificing its fast computation time. Concerning the ensemble averaging of SGNNs, 10-20 ensembles may be appropriate judging from our experiments. Though this ensemble averaging model sacrifices its computing time, this problem can be solved by considering parallel processing. Flom above mentioned facts, the SGNT approach is one of the powerful neurar network algorithm and has a generalization capability competable with most widely used BP algorithm in feed-forward neural networks.

References

1. Wen, W. X., Pang, V. and Jennings, A.: Self-Generating vs. Self-Organizing, What's Different? In: Simpson, P. K. (eds.): Neural Networks Theory, Technology, and Applications. IEEE, New York (1996) 210-214
2. Kohonen, T.: Self-Organizing Maps. Springer-Verlag, Berlin (1995)
3. Wen, W. X., Jennings, A. and Liu, H.: Learning a Neural Tree. In: International Joint Conf. on Neural Networks. Beijing (1992) 751-756
4. Inoue, H. and Narihisa, H.: Performance of Self-Generating Neural Network Applied to Pattern Recognition. In: 5th International Conf. on Information Systems Analysis and Synthesis, Vol. 5. Orlando, FL (1999) 608-614
5. Rumelhart, D., Hinton, G. E. and Williams, R. J.: Learning Internal Representations by Error Propagation. In: Rumelhart, D., McClelland, J. and the PDP Reserch Group (eds.): Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambridge, MA (1986) 318-362
6. Thrun, S. B. et alt: The MONK's Problems – A Performance Comparison of Different Learning Algorithms. Technical report CMU-CS-91-197. Carnegie Mellon University (1991)
7. Prehelt, L.: PROBEN1 – A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical report 21/94. Universität Karlsruhe (1994)

Attribute Transformations on Numerical Databases

Applications to Stock Market and Economic Data

Tsau Young Lin[1,2] and Joseph Tremba[1]

[1] Department of Mathematics and Computer Science
San Jose State University,
San Jose, California 95192
tylin@cs.sjsu.edu
[2] Berkeley Initiative in Soft Computing
Department of Electrical Engineering and Computer Science
University of California,
Berkeley, California 94720
tylin@cs.berkely.edu

Abstract. The effects of attribute transformations on numerical data mining are investigated. Theoretical examples from classical mathematics are used to illustrate its critical-ness. The simplest kind of attribution transformations, linear transformations, is applied to stock market and economic data. Some useful "predictive" rules are generated. Here "predictive" is used in the sense that the logical patterns involve time elements.

Keywords: database, stock market data, rough set, extensional databases, predictive

1 Introduction

Intuitively, it is clear that selecting suitable attributes is often the key to the success of data mining. In AI, one often uses knowledge representations to describe an unknown universe by known features. So the selected features are often suitable for knowledge discovery. However in database, the attributes are often selected for totally different purposes; they are selected primary for record keeping. So the existing attributes might not be suitable for data mining; attribute transformations are often needed.

This is our first paper to investigate attribute transformations systematically. First we look at linear transformations of attributes. The study is reasonably successful. Linear transformations are well understood mathematics and intuitively reachable areas. The results are applied to stock and economic data. Even in such simple applications, experiments are needed: Should we take sums, averages or weighted averages, and for three days, four days, ..., n-days, and etc.? Some understanding of the data seems necessary.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 181–192, 2000.
© Springer-Verlag Berlin Heidelberg 2000

Next, we turn to the non-linear transformations. This is a much tougher and deeper area. Many data mining of daily life problems probably would not need such sophisticated transformations, however, the area of scientific discoveries may need such transformations, simply because some interesting scientific data (attribute) are often not directly observable and measurable. To search for guidance, some classical mathematics is examined. We find examples that indicate clearly attribute transformations are a "must," at the same time there are no clear hints on how such transformations can be constructed; see Section 3.

Finally, we observe that attribute transformations can always be approximated by polynomials. So in the case no meaningful transformations can be recommended by domain experts, the polynomials of attribute transformations may be searched brutally by one degree at a time. Attribute transformations, we believe, will become, if have not been, one of the most important areas of data mining research.

The data mining methodology we used in this paper is based on rough set theory [5]. Rough set theory has two formats, abstract and table formats [3]. In the table format, it is a theory of extensional relational databases ([1] pp. 90). However, rough set theory and the traditional theory of extensional relational database are fundamentally different. The latter focuses on storing and retrieving data, while the former on summarizing the patterns or rules [2] - a subset of modern data mining theory.

2 Information Tables - Rough Set Theory

Here we present the table format of rough set theory. Roughly, both relational and rough set theories are studies of attribute-value-pair representations of the universe. A relation in the database is the image of such a knowledge representation, while information table is the graph of such a representation.

An information table (also known as information system, knowledge representation system) consists of

1. $U = \{u, v, \ldots\}$ is a set of entities.
2. A is a set of attributes $\{A^1, A^2, \ldots A^n\}$.
3. $dom(A^i)$ is the set of the values of the attribute A^i

$$Dom = dom(A^1) \times dom(A^2) \ldots \times dom(A^n)$$

4. $\rho : U \times A \longrightarrow Dom$, called description function, is a map such that

$$\rho(u, A^i) \text{ is in } dom(A^i) \text{ for all } u \in U \text{ and } A^i \in A.$$

The description function ρ induces a set of maps

$$t = \rho(u, -) : A \longrightarrow Dom.$$

Each image forms a tuple:

$$t = (\rho(u, A^1), \rho(u, A^2), ..., \rho(u, A^i), ..\rho(u, A^n))$$

Note that each tuple t is associated with an object u, but not necessarily uniquely. In an information table, two distinct objects could have the same tuple representation that is not permissible in relational databases.

A decision table is an information table (V, A, Dom, ρ) in which the attribute set $A = C \cup D$ is a union of two non-empty sets, $C = \{C^1, C^2, \ldots C^p\}$ and $D = \{D^1, D^2, \ldots D^q\}$, of attributes. The elements in C are called conditional attributes, D are decision attributes. Each tuple can be regarded as a decision rule. Rough set theory, then, provides a method to reduce a given decision table to a minimal table, called redut, which consists of a minimal collection of the simplest decision rules; one should note that a given decision table may have several reducts [5].

Strictly speaking, rough set theory is more of data reduction than data mining. The reduct is logically equivalence to the original decision table in the sense that both the given decision table and its reduct will make the same decision. It may be somewhat a surprise to rough set community, L. Zadeh reached a similar conclusion in 1976 ([8] pp278); he called it compactification of a tabular representation.

3 Attribute Transformations

In this section, we begin our study of attribute transformations systematically. We examine, transpose, linear, quadratic, and other nonlinear transformations. The study conclude that attribute transformations are often necessary. However, there are no indications on how such transformations could be found. In practice, such transformations most likely will be provided by domain experts. Finally, we observe from elementary algebra that each attribute transformation is a polynomial function; So a brutal force search of a proper transformation (by one degree at a time) is *costly* possible. We believe the understanding of the attribute transformations will soon, if have not been, be one of the major areas of data mining research.

3.1 Time Series-Transpose/Delay Operations

This is an intuitively meaningful operation; we create a table of time series. The original table, which consists of first three columns, A1, B1 and C1, is reproduced six additional times. Each time it is reproduced, the columns or objects are shifted up by one row. The 1st group of columns is the records starting from "first day" record (the original table), 2nd group "second day" (one unit time delay) and etc. The attribute name is appended with a number indicating the number of delays. In Table 1, we are showing six delays. The new table is seven times wider than the original table.

Table 1. shows how data is delayed with each of the six reproductions of the original table.

X	A_1	B_1	C_1	A_2	B_2	C_2	...	A_7	B_7	A_7
1	1	1	1	2	2	2		7	7	7
2	2	2	2	3	3	3		8	8	8
3	3	3	3	4	4	4		:	:	:
4	4	4	4	5	5	5				
5	5	5	5	6	6	6				
6	6	6	6	7	7	7				
7	7	7	7	8	8	8				
8	8	8	8	:	:	:				

3.2 Linear Transformations

In this section, we investigate a class of transformations that are mathematically well understood. Some linear transformations, such as sums, averages, rotations, translations and etc. are intuitively meaningful. Let $B = \{B_1, B_2, \ldots B_m\}$ be a subset of the attributes $A = \{A_1, A_2, \ldots A_n\}$. Let $r = \{r_1, r_2, \ldots r_m\}$ be a set of real numbers. We form a new derived attribute A^* by taking a linear combination of attributes in B, namely,

$$A^* = r_1 B_1 + r_2 B_2, .. + r_m B_m$$

In Table 2, we give an interesting special case, averaging, that is, we set $r_1 = \ldots = r_m = 1/5$:

$$A^* = (A1 + A2 + A3 + A4 + A5)/5;$$
$$B* = (B1 + B2 + B3 + B4 + B5)/5;$$
$$C* = (C1 + C2 + C3 + C4 + C5)/5.$$

Table 2. shows the combined effects of delay and averaging.

X	A_1	B_1	C_1	...	A_5	B_5	C_5	...	$A*$	$B*$	$C*$
1	1	1	1		5	5	5		$(1+2+3+4+5)/5$		
2	2	2	2		6	6	6		$(2+3+4+5+6)/5$		
3	3	3	3		7	7	7		$(3+4+5+6+7)/5$		
4	4	4	4		8	8	8		$(4+5+6+7+8)/5$		
5	5	5	5								
6	6	6	6								
7	7	7	7								
8	8	8	8								

3.3 Quadratic Transformations

Next, we will consider quadratic transformations, namely,

$$B^* = r_1(B_1)^2 + r_2(B_2)^2, ..., +r_m(B_m)^2 + r_{(1,2)}B_1B_2 +, ..., +r_{((m-1),m)}B_{(m-1)}B_m$$

where $r_i, r_{(j,h)}$ are real numbers. This is another class of transformations, which is reasonably understood mathematically. First, we will show their effects from classical mathematics: The first six columns of Table 3 consists of the coefficients of the equations of some conic sections:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

It is clear that we have very little to say about these tuples if we only view the first six attributes. So, we compute a quadratic transformation, called discriminant,

$$A^* = (B^2 - 4AC)$$

and consider the signs of its values. The sign classifies (generalizes) the values of the discrimnant into three concepts, positive, negative and zero; denoted by +, -, 0. These signs have standard meaning; see the last column in Table 3. It is clear that without the new derived attributes A* and the sign, we will not be able to classifies the conic sections. This example indicates that some proper transformations are needed for data mining, unfortunately, it gives us no hints how such transformations can be discovered. Nevertheless, since quadratic transformations are relatively simple, there is a possibility that the right transformation can be found by brutal force experiments.

Table 3. shows the effects of quadratic transformations

A	B	C	D	E	F	$A*$	*Sign*	*Interpretations*
9	−4	−72	0	8	176	2608	+	*Ellipse*
9	0	16	0	0	−144	−576	−	*Hyperbola*
73	72	52	30	−40	−75	−10000	−	*Hyperbola*
2	−72	23	−80	−60	−125	5000	+	*Ellipse*
1	0	0	0	4	0	0	0	*Parabola*
1	0	0	12	−1	39	0	0	*Parabola*

3.4 Geometric Considerations

We will consider a geometric example, the first six columns of Table 4 are the (X, Y)-coordinates of three vertices of triangles. The derived attributes are functions, namely, the length of three segments, of the original attributes. To discover any theorems about these triangles, the original attributes are not useful at all. So the following three new derived attributes, the distances, are considered

$$A^* = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2},$$
$$B^* = \sqrt{(X_2 - X_3)^2 + (Y_2 - Y_3)^2},$$
$$C^* = \sqrt{(X_3 - X_1)^2 + (Y_3 - Y_1)^2}.$$

It is clearly these new attributes of Table 4 show that all segments are the same length. Such a conclusion is clearly impossible from the original attributes. This is another example saying that some specific attribute transformations are often a must, and can only be provided by domain experts.

Table 4. shows the effects of geometric transformations

X_1	Y_1	X_2	Y_2	X_3	Y_3	A^*	B^*	C^*
0	1	−1	0	1	0	$\sqrt{2}$	2	$\sqrt{2}$
1	2	0	1	2	1	$\sqrt{2}$	2	$\sqrt{2}$
$\sqrt{3}/2$	1/2	$-1/2$	$\sqrt{3}/2$	1/2	$-\sqrt{3}/2$	$\sqrt{2}$	2	$\sqrt{2}$
−2	1	−1	0	−1	2	$\sqrt{2}$	2	$\sqrt{2}$

At this point, we can offer the following geometric observations. A numerical n-relation (degree n relation) is a finite set of points that is lying in a "hyper surfaces"(manifold) of an n-dimensional Euclidean space. Each set of attribute selections corresponds to a selection of coordinate systems. To have nice qualitative information about such a set of points, there is a need of a specific coordinate system. In other words, each problem requires its own set of transformations. So we conclude that data mining often needs a suitable transformation, and the selection of the transformation is dictated by the specific nature of the given problem.

3.5 Polynomial Approximation of Transformations

Let us conclude this section with a general formulation of attribute transformations. Let $A, B, ..., X$, be a set of given attributes. We often need to compute a derived attribute (see Table 5), which is a function of the original attributes

$$FY = f(FX_1, FX_2, \ldots, FX_n).$$

Since the table is of finite size, the function f can take polynomial forms, namely,

Theorem f is a polynomial function.

Proof: The columns from A to X are the conditional attributes, and FY is the decision attribute. Each row of the conditional attributes can be regarded as a point x in Euclidean space, and the corresponding decision value is a value y assigned to x. For convenience, we will call these points x conditional points; there are as many points as the table size. By a simple result of the college algebra, there is a polynomial function f which assumes the value y at each conditional point x, that is, $y = f(x)\ \forall\ x$. Q.E.D.

By this theorem, we note that the distance functions in Table 4 can be expressed by polynomials. Mathematically, these polynomials are the Weistrass Approximation of the distance functions on the conditional points. These distance polynomials are not intrinsic in the sense the "forms" of these polynomials are *table dependent*. In other words, the polynomials will vary as the table changes. Note that the "form" of the original distance functions are table independent, they will not change as the table varies.

Table 5. shows the effects of general transformations

A	B	C	...	X	FY
a_1	b_1	c_1	...	x_1	$f_1 = f(a_1, b_1, , ..., x_1)$
a_2	b_2	c_2	...	x_2	$f_2 = f(a_2, b_2, , ..., x_2)$
a_3	b_3	c_3	...	x_3	$f_3 = f(a_3, b_3, , ..., x_3)$
...	...	...	...	...	...
a_i	b_i	c_i		x_i	$f_i = f(a_i, b_i, , ..., x_i)$

4 Applications to Economic and Stock Market Data

Our study of attribute transformations is just in the beginning. However, the linear case is reasonably complete. We will apply it to the stock market and economic data. The rest of the paper is extracted from [7]. A six year stock data range of August 1, 1990 to July 31, 1996 was selected (including some indices [9], [10]. This provided 1518 cases for the daily experiment. The next step is to find rules about the stock price of the company Applied Materials. The rule-generation program DataLogic/R+ is used [6]. The decision attribute is the stock price of Applied Materials.

4.1 Data Selection

The daily experiment attributes selected were in Table 6:

4.2 Data Preparation

All the attribute data for the daily experiments was discretized into integer values. The real-number closing price percentage change was rounded to the nearest integer value. The basic decision table had 1517 rows and 11 columns. Next the three programs for delaying, averaging, and summing were run on this table thus producing three new tables each with 77 columns and 1511, 1487, and 1511 rows respectively; the detail of the three programs can be found in [7]. The decision attribute was generalized; see Table 7. The results provide strong predictive rules hopefully that provide some insight into the direction of the change but no information about the amount of the change.

Table 6. Data Selected

Condition Attribute	Type	Symbol
Dow Jones Industrial Average	Index	Dow
Compaq Computer	Stock	CPQ
IBM	Stock	IBM
Digital Equipment	Stock	DEC
Intel	Stock	Intel
Motorola	Stock	Mot
Texas Instruments	Stock	TI
Semiconductor Index	Index	Semi
Electronics Index	Index	Elec
Electronic Equipment Index	Index	Equip
Decision Attribute	*Type*	*Symbol*
Applied Materials (daily price)	Stock	Applied

Table 7. Generalization

Classification	High lever concept
$x < 0.5$	Falling
$-0.5 \leq x0.5$	No Change
$x \geq$	Rising

5 Analysis of Results

Preliminary experiments were tried with the decision attribute being an integer value representing the percentage change from the previous case. The daily percentage change ranged from -51 to + 17 for the decision attribute. This would provide 68 classifications or concepts to be evaluated. The resulting rules were weak and possessed only a few supporting cases. This led to the choice of allowing only three classifications. The condition attributes remained unchanged as integer values. The rules generated for the daily and monthly experiments will now be listed.

5.1 Daily Rules

The three programs were run in the rule generator program for the decision attribute of Applied Materials daily stock price discretized into three categories of falling, no change, and rising. The rules developed for each of the categories are:

1. Falling Rules
   (a) $[EQUIP1 < 1]\&[TI1 < -2]\&[-4 \leq SEMI1 \leq -1] \rightarrow [APPLIED = -1]$

Stock price will fall if the 5 day average of EQUIP is less than 1% and the 5 day average of TI is less than -2% and the 5 day average of SEMI is between -4% and -1% (source: average program). This rule is supported by 62.2% of the 254 cases.

(b) $[-20 \leq CPQ1 \leq -13] \rightarrow [APPLIED = -1]$
Stock price will fall if yesterday CPQ was between -20% and -13% (source: delay program). This rule is supported by 100% of the 2 cases.

(c) $[-10 \leq EQUIP2 \leq -1] \rightarrow [APPLIED = -1]$
Stock price will fall if the two day sum of EQUIP was between -10% and -1% (source: sum program). This rule is supported by 65.5% of the 554 cases.

2. No Change Rules

(a) $[-2 \leq TI1 \leq 1]\&[1 \leq DEC1 \leq 2]\&[MOT2 \geq 2] \rightarrow [APPLIED = 0]$
Stock price will not change if the 5 day average of TI is between -2 % and 1 % and the 5 day average of DEC is between 1 % and 2 % and the 10 day average of MOT is greater than or equal to 2 % (source: average program). This rule is supported by 66.7 % of the 3 cases.

3. Rising Rules

(a) $[EQUIP1 < -3 or EQUIP1 > 0]\&[TI1 \geq 1] \rightarrow [APPLIED = 1]$
Stock price will rise if the 5 day average of EQUIP is either less than 3% or greater than 0% and the 5 day average of TI is greater than or equal to 1% (source: average program). This rule is supported by 62.5 % of the 253 cases.

(b) $[SEMI1 < -4 or SEMI1 > -1]\&[-3 \leq EQUIP1 \leq 0]\&[TI1 \leq -3 or TI1 \geq 4] \rightarrow [APPLIED = 1]$
Stock price will rise if the 5 day average of SEMI is either less than -4% or greater than -1% and the 5 day average of EQUIP is between -3% and 0% and the 5 day average of TI is either less than or equal to -3% or greater than or equal to 4% (source: average program). This rule is supported by 100% of the 3 cases.

(c) $[-4 \leq SEMI1 \leq -1]\&[TI1 \geq 1] \rightarrow [APPLIED = 1]$
Stock price will rise if the 5 day average of SEMI is between -4% and -1% and the 5 day average of TI is greater than or equal to 1% (source: average program). This rule is supported by 77.8% of the 9 cases.

(d) $[-8 \leq IBM1 \leq -5]\&[MOT5 = 1] \rightarrow [APPLIED = 1]$
Stock price will rise if yesterday IBM is between -8% and -5% and 5 days ago MOT was equal to 1% (source: delay program). This rule is supported by 100% of 4 cases.

(e) $[-2 \leq IBM6 \leq 0]\&[-8 \leq IBM1 \leq -4]\&[MOT5 \leq -1 or MOT5 \geq 2] \rightarrow [APPLIED = 1]$
Stock price will rise if 6 days ago IBM is between -2% and 0% and yesterday IBM was between -8% and -4% and 5 days ago MOT was either less than or equal to -1% or greater than or equal to 2% (source: delay program). This rule is supported by 100% of the 3 cases.

(f) $[IBM6 < -2 or IBM6 > 0]\&[IBM1 \leq -7 or IBM1 \geq 1]\&[MOT5 \leq -2 or MOT5 \geq 2] \rightarrow [APPLIED = 1]$

Stock price will rise if 6 days ago IBM was either less than -2% or greater than 0% and yesterday IBM was either less than or equal to -7% or greater than or equal to 1% and 5 days ago MOT was either less than or equal to -2% or greater than or equal to 2% (source: delay program). This rule is supported by 60% of the 180 cases.

(g) $[EQUIP2 \geq 1] \rightarrow [APPLIED = 1]$
Stock price will rise if the two day sum of EQUIP is greater than or equal to 1% (source: sum program). This rule is supported by 65.9% of the 627 cases.

5.2 Daily Rule Validation

The rules are validated by comparing them against recent test data and determining the percentage of correct predictions. The daily test data used the eight month period from August 1, 1996 to April 9, 1997. The monthly test data used the five month period from August 1, 1996 to December 31, 1996. The period could not be made larger because the SIA index was changed in January, 1997 [28]. The raw data was preprocessed using the same programs to produce the delayed, averaged, and cumulative tables. The tables were reviewed manually to determine the rule compliance. The spreadsheet program Excel by Microsoft was used as an aid to sort the table based on rule attribute values. The following table (Table 8 lists for each rule the number of cases found, the percent of correct cases, the number of cases found during rule generation, and the percent of correct cases during rule generation:

Table 8. Rules Analysis

Rule No	No of Cases	Correct	Learning %	Learning %
1.	0	*NA*	254	62.2
2.	0	*NA*	2	100
3.	58	70.7	554	65.5
4.	3	0.0	3	66.7
5.	46	60.8	253	62.5
6.	0	*NA*	3	100
7.	2	50.0	9	77.8
8.	0	*NA*	4	100
9.	1	0.0	3	100
10.	19	68.4	180	60.0
11.	72	80.5	627	65.9

The daily validation results can be separated into two groups. The first group, consisting of rules 4 and 7, includes those rules that came from a small number of supporting cases (less than 10). In this group, the error rates were very high.

In the second group consisting of rules generated with a large number of supporting cases (rules 3, 5, 10, and 11), the percentage correct was very close to the percentage from the learning experiment. The differences ranged from -1.7 % to 14.6 %. Rules 1, 2, 6, and 8 had no instances in the validation experiment.

5.3 Conclusions on Stock and Economic Data Analysis

For the stock market data analysis, the following conclusions were reached:

1. The combined use of delay, average, and cumulative preprocessing of data are useful tools for analyzing time series data.
2. The ratio of rising to falling rules derived was 2.3 to 1. This is believed to have been due to the dominant bull market occurring during the six year learning period.
3. When working with continuous or semi-continuous data such as stock market data, rough set theory does not provide a means for selecting or optimizing the discretizing ranges. It was through trail and error that the ranges used in this paper were chosen.

6 Conclusions

Here are some of our conclusions:

1. In some data mining problems, attribute transformations are necessary (Section 3.4, 3.3).
2. Rough set theory can be used as a data mining tool for the new tables. (Section 2).
3. Meaningful attribute transformations should be suggested by domain experts.
4. Since information table is of finite size, attribute transformations can be approximated by polynomial functions (Section 3.5).
5. Hence brutal force search might be employed (one degree at a time), if the domain experts can not suggest a meaningful transformation (Sections 3.2, 3.3).
6. Linear transformations have proved to be useful in stock market and economic data (Section 5.3).
7. Even in such a case, some light brutal force searches have been employed; should 2 day, 3 day ,... n day average be used? (Section 5.3, Item 3)

References

1. Date, C. J.: Introduction to Database Systems. 3rd,7th edn. Addision-Wesely, Reading, Massachusetts (1981, 2000).
2. Lin, T. Y.: An Overview of Rough Set Theory from the Point of View of Relational Databases, Bulletin of International Rough Set Society, vol. 1, no. 1 (1997) 30-34.
3. Lin, T. Y.: Guest Editorial, Intelligent Automation and Soft Computing, an International Journal, Vol 2, No 2 (1996) 94-94.

4. Meyer,D.: The Theory of Relational Databases. Computer Science Press, 1983 (6th printing 1988).
5. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data, Kluwer Academic, Dordrecht (1991)
6. Software: DataLogic/R and DataLogic/R+ Rough Sets Guide. version 1.3, Reduct and Lobbe, Regina, Saskatchewan, Canada (1993).
7. Tremba, J.: An Application of Rough Sets to Economic and Stock Market Data, thesis, San Jose State University, (1997).
8. Zadeh, L.: A fuzzy-algorithmic approach to the definition of complex and imprecise concepts, J. Man-machine Studies 8 (1976) 249-191.
9. Semiconductor Industry Association, Seasonally Adjusted Book-To-Bill Ratios, San Jose, CA.
10. Semiconductor Equipment and Materials International, Market Statistics, Historical Book to Bill, Mountain View, CA, 1996.

Tsau Young (T. Y.) Lin received his Ph. D from Yale University, and now is a Professor at Department of Mathematics and Computer Science, San Jose State University, the Metropolitan University of Silicon Valley, also a BISC fellow at the University of California-Berkeley. He has served as editors, associate editors, members of advisory or editorial board of several international journals, and chairs, co-chairs and members of program committees of conferences. His interests include approximation retrievals, data mining, data warehouse, data security, and new computing methodology (granular, rough, and soft computing).

Joseph Tremba is a Software Manager at Applied Materials. Tremba received a BS and MSEE from the University of Michigan, an MBA from Michigan State University, and a MSCS degree from San Jose State University. He is a member of IEEE and Eta Kappa Nu. Contact him at joe_trembaamat.com.

Efficient Detection of Local Interactions in the Cascade Model

Takashi Okada

Kwansei Gakuin University, Center for Information & Media Studies
Uegahara 1-1-155, Nishinomiya
662-8501 Japan
okada@kwansei.ac.jp

Abstract. Detection of interactions among data items constitutes an essential part of knowledge discovery. The cascade model is a rule induction methodology using levelwise expansion of a lattice. It can detect positive and negative interactions using the sum of squares criterion for categorical data. An attribute-value pair is expressed as an item, and the *BSS* (between-groups sum of squares) value along a link in the itemset lattice indicates the strength of interaction among item pairs. A link with a strong interaction is represented as a rule. Items on the node constitute the left-hand side (LHS) of a rule, and the right-hand side (RHS) displays veiled items with strong interactions with the added item. This implies that we do not need to generate an itemset containing the RHS items to get a rule. This property enables effective rule induction. That is, rule links can be dynamically detected during the generation of a lattice. Furthermore, the *BSS* value of the added attribute gives an upper bound to those of other attributes along the link. This property gives us an effective pruning method for the itemset lattice. The method was implemented as the software DISCAS. There, the items to appear in the LHS and RHS are easily controlled by input parameters. Its algorithms are depicted and an application is provided as an illustrative example.

Keywords: local interaction, cascade model, sum of squares, itemset lattice, pruning of lattice.

1 Introduction

Itemset representation, first introduced in association rule mining [1], offers a flexible and uniform framework for a learning task. Both classification and characteristic rules have been induced using this framework [2,•3]. Bayesian networks and Nearest Neighbor classifiers were also formulated as the mining of labeled itemsets [4].

Detection of local interactions is necessary to obtain valuable knowledge from the itemset lattice. Here, the term "local interaction" is used in two ways. Firstly, it shows that some value pairs of two attributes are correlated. For example, two attributes: A and B indicate strong interactions at row [A:•a 3] and at column [B:•b3] in the contingency table on the next page, while minor interactions are found in other cells.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 193-203, 2000.
© Springer-Verlag Berlin Heidelberg 2000

Secondly, "local" denotes that an interaction appears when some preconditions are satisfied. The interactions in Table 1 may appear only in the cases with [C:•c1].

Table 1. Example of a local interaction between attributes A, B. Each cell shows the distribution of 30 cases with [C: •c1] item

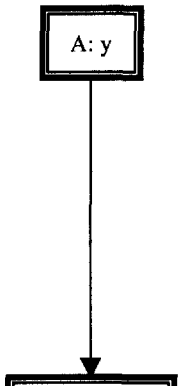
in 30 cases with [C: c1]	[B: b1]	[B: b2]	[B: b3]
[A: a1]	5	5	0
[A: a2]	5	5	0
[A: a3]	0	0	10

Silverstein et al. succeeded in detecting interactions using χ^2 test based on levelwise lattice expansion [5]. They showed the importance of local interactions between value pairs. Their formulation enabled the detection of a negative interaction that was missed by association rule mining. The problem of lattice explosion was also solved by the upward-closed property of dependency in the lattice, and hence the method was very fast. However, their formulation did not detect the simultaneous occurrence of plural interactions. As a result, it required difficult speculations to find such rules as "IF [A: •a1] THEN [B:•b 2, C:•c 3]". What is needed is the ability to detect interactions among plural attributes in the lattice and compare the strengths of these interactions.

The authors previously proposed the cascade model as a framework for rule induction [6], and subsequently showed that the sum of squares (*SS*) criterion for categorical data gave a reasonable measure of the strength of the interaction when we partitioned a dataset by the values of an attribute [7]. Detailed descriptions of *SS* properties for categorical data have been published separately [8]. In this paper, our focus is on an efficient and effective method of rule mining in the cascade model. The next section gives a brief introduction to the model and the underlying *SS* criterion. Section 3 describes an efficient method for detecting local interactions. The results of applying the method to House voting-records are discussed in Sect. 4.

2 The Cascade Model

The cascade model examines the itemset lattice where an [attribute:•value] pair is employed as an item to constitute itemsets. Links in the lattice are selected and expressed as rules. Figure 1 shows a typical example of a link and its rule expression. Here, the problem contains five attributes: A• –•E , each of which takes (y, n) values. The itemset at the upper end of the link has an item [A:•y], and another item [B:•y] is added along the link. Items of the other attributes are called veiled items. Three small tables at the center show frequencies of the items veiled at the upper node. The corresponding *WSS* (within-group sum of squares) and *BSS* (between-groups sum of squares) values are also shown along with their sample variances. Following the variance definition of a categorical variable [9], WSS_i and BSS_i were given by the following formulae [7],



	y	n	*WSS*	σ^2
B	60 (9.6)	40 (14.4)	24.0	.24
C	50 (12.5)	50 (12.5)	25.0	.25
D	60 (9.6)	40 (14.4)	24.0	.24
E	40 (14.4)	60 (9.6)	24.0	.24

	BSS
B	9.60
C	0.00
D	6.67
E	5.40

IF [B: y] added on [A: y]
THEN [D: y; E: n]
Cases: 100 - 60
[D: y] 60% - 93%, BSS = 6.67
[E: n] 60% - 90%, BSS = 5.40

	y	n	*WSS*	σ^2
B	60 (0.00)	0 (0.00)	0.00	.000
C	30 (7.50)	30 (7.50)	15.00	.250
D	56 (0.25)	4 (3.48)	3.73	.062
E	6 (4.86)	54 (0.54)	5.40	.090

Fig. 1. A sample link, its rule expression and properties of the veiled items. See Sect. 3.2 for the explanation of values in parentheses.

$$WSS_i = \frac{n}{2}\left(1 - \sum_a p_i(a)^2\right) \quad , \tag{1}$$

$$BSS_i = \frac{n^{\mathrm{L}}}{2}\sum_a \left(p_i^{\mathrm{L}}(a) - p_i^{\mathrm{U}}(a)\right)^2 \quad , \tag{2}$$

where i designates an attribute, and the superscripts U and L are attached to show the upper and the lower nodes, n shows the number of supporting cases of a node, and $p_i(a)$ is the probability of obtaining the value a for the attribute i.

A large BSS_i value is evidence of a strong interaction between the added item and attribute i. The textbox at the right in Fig. 1 shows the derived rule. The added item [B: y] appears as the main condition in the LHS, while the items on the upper node are placed at the end of the LHS as preconditions. When a veiled attribute has a large BSS_i value, one of its items is placed in the RHS of a rule. An item selection method from a veiled attribute was described in [7].

We can control the appearance of attributes in the LHS by restricting attributes in the itemset node. On the other hand, the attributes in the RHS can be selected by setting the minimum BSS_i value of a rule ($min\text{-}BSS_i$) for each attribute. The cascade model does not exclude the possibility of employing a rule link between distant node pairs if they are partially ordered to each other in the lattice. The main component of the LHS may then contain plural items, though we cannot compare the advantages of flexibility of expression to the disadvantages of increased computation time. Either way, items in the RHS of a rule are not necessary for them to reside in the lattice. This is in sharp contrast to association rule miners, which require the itemset, [A: y; B: y; D: y; E: n] to derive the rule in Fig. 1.

3 Methods

Since association rule mining was first proposed, a great deal of research effort has been directed towards finding effective methods of levelwise lattice generation [10, 11, 12]. However, vast amounts of computation are still necessary. When we handle table data, dense items result in a huge number of itemsets at the middle level of the itemset lattice. In this section, we first propose a new algorithm for rule induction. We then discuss the problem of lattice pruning and the control of rule expressions.

3.1 Basic Mechanism

The previous section described that a rule description is possible if the LHS items appear as an itemset node in a lattice and if the frequencies of the veiled items are known. We then immediately notice that the following two procedures can be used to improve the rule induction process.

- *No* Apriori *condition check.* We can use the frequency information of the veiled items at the node generation step. That is, items satisfying the minimum support condition are selected to make new nodes. We can discard an item whose count is lower than the minimum support. For example, if the minimum support is set to 10 in Fig. 1, four new nodes, made by the addition of items: [C:•y], [C:•n], [D:•y] and [E: • n] to the lower node, are necessary and sufficient.
- *Dynamic detection of rule links.* Before the entire lattice is constructed, we can detect strong interactions and send the relevant link to another process that extracts rules and provides them for real-time operations. As strong interactions with many supports are expected to appear in the upper part of the lattice, this will give us a practical way to implement OLAP and to mine valuable rules from a huge dataset.

The above points are realized as the algorithm CASC, shown in Fig. 2. In this algorithm, *nodes*(L) shows the set of itemset nodes at the L-th level of the lattice. After creating the root-node with no items and counting all items in the database, *create-lattice* expands the lattice in a levelwise way, changing the lattice level L. In each lattice level, it counts the veiled items and detects interactions. Then *generate-next-level* simply makes nodes following the first procedure. Section 3.2 discusses a new *pruning-condition* added to the minimum support. The second procedure is implemented as *detect-interactions*, which compares two nodes in the L-th and (L+1)-th levels. Hashing is used to fetch the upper node quickly. If a node pair has a veiled attribute for which BSS_i exceeds the given *min-*BSS_i parameter, then the function sends it to another process. The last function, *count*, is the most time consuming step. The subset relationship between the items in a case and those in a node is judged using the trie data structure. If the condition holds, the count of the veiled items on the node is incremented.

Here, we note that an upper node does not always exist in the process of *detect-interactions*, as we do not use the *Apriori* condition in the node generation step.

```
create-lattice()
  nodes(0) := {root-node}
  count(nodes(0) database)
  loop changing L from 1 until null(nodes(L))
    nodes(L) := generate-next-level(nodes(L-1))
    count(nodes(L) database)
    detect-interactions(nodes(L))

generate-next-level(nodes)
  loop for node in nodes
    loop for item in veiled-items(node)
      if pruning-condition is not applicable
        push make-new-node(item node) to new-nodes
  return new-nodes

detect-interactions(lower-nodes)
  loop for node in lower-nodes
    loop for itemset in omit-one-item(node)
      upper := get-node(itemset)
      if for some i, BSSi(node upper)>min-BSSi then
        send-link(node upper)

count(nodes database)
  loop for case in database
    loop for node in nodes
      if itemset(node)⊂items(case) then
        increment item-count(node) for items(case)
```

Fig. 2. Algorithm CASC

3.2 Pruning Lattice

The idea of pruning is clear if we think of adding a virtual attribute, B': a copy of B, in the example provided by Fig. 1. When we generate a new node adding the item [B': •y] under the lower node, it gives us nothing, as all frequencies remain the same. Note that the interactions between B' and (D, E) are detected separately on another node. Even if the correlation is not so complete as that between B and B', we might prune new links that add highly correlated attributes like D and E in Fig. 1.

Suppose there is a link between nodes U and L. U has veiled attributes $\{x_i\}$ and L is a descendent node of U added by an item, $[x_0{:}\bullet a_0]$. We employed the following inequality to prune the link between U and L. A proof of this inequality is given in the Appendix.

$$\begin{aligned} BSS_i \le (m_i/2) \bullet\bullet BSS_0 &= (m_i/2) \bullet\bullet n^{L} \bullet\bullet (1 - p_0{}^{U}(a_0))^2 \\ &= (m_i/2) \bullet\bullet n^{U} \bullet\bullet p_0{}^{U}(a_0) \bullet\bullet (1 - p_0{}^{U}(a_0))^2 \end{aligned} \tag{3}$$

BSS_i denotes the BSS value for a veiled attribute x_i between U and L, $p_0^U(a_0)$ is the probability of attribute x_0 having the value a_0 at node U, and m_i denotes the number of attribute values of x_i.

Our objective is to find links with large BSS_i values. Suppose that the threshold of the BSS_i value for the output rule is set to $N \bullet\bullet\bullet t\ h r e s$, where N is the total number of cases and *thres* is a user-specified parameter. Then, the above inequality implies that we do not need to generate the link U–L, if the RHS of (3) is lower than $N \bullet\bullet\bullet th\ r e s$. This pruning condition is written as,

$$(m_i/2) \bullet n^U \bullet p_0^U(a_0) \bullet (1 - p_0^U(a_0))^2 < N \bullet\bullet \bullet thres \quad . \tag{4}$$

If all possible RHS attributes are assigned the same *min-BSS*$_i$, *thres* can be set to *min-BSS/N*. The LHS of (4) takes the highest value at $p_0^U(a_0) = 1/3$. Then, if n^U is small, we can prune the lattice for a wide range of $p_0^U(a_0)$ values at a given $N \bullet thres$. On the other hand, if n^U is large, then the pruning is limited to those links with $p_0^U(a_0)$ values far from 1/3. The tables attached at the nodes in Fig. 1 show these LHS values of (4) in parentheses. Suppose that N is 400 and *thres* is 0.01. Then the meaningful branches of the lower node are limited to those links by the addition of three items, [C:•y], [C: •n] and [E: •y].

Lastly, we have to note the properties of this pruning strategy. There is always the possibility of other local interactions below the pruned branch. For example, if we prune the branch from [A: •y, B:•y] to [A:•y, B:•y, D:•y], there might be an interaction between [C: •n] and [E:•n] under the pruned node, as shown in Fig. 3. However, we can expect to find the same kind of interaction under the node [A:•y, B: •y] unless the interaction is truly local on the lower pruned node. The upper rule in Fig. 3 covers broader cases than the lower rule does. So, we call this upper rule a *broader relative rule* of the lower pruned rule.

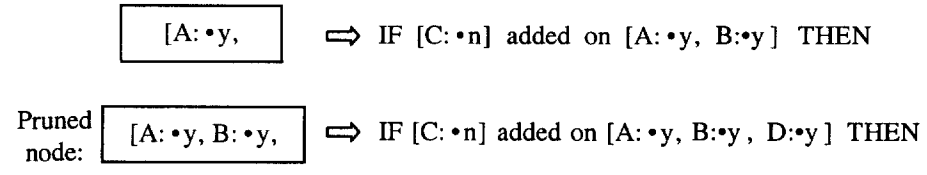


Fig. 3. A pruned rule and its broader relative rule

3.3 Symmetric and Concise Control in Rule Generation

Two input parameters, *min-BSS*$_i$ and *thres*, affect rule expression. A very high *min-BSS*$_i$ value excludes the attribute x_i from the RHS of the rules. Suppose that the pruning condition (4) is extended to use *thres*$_i$ for each attribute x_i. Then, we can prohibit the attribute x_i from entering the LHS of a rule if we give *thres*$_i$ a very high value.

Setting a high *thres*$_i$ value to the class attribute and high *min-BSS*$_i$ values to the explanation attributes results in discrimination rules. On the other hand, setting

affordable values to these parameters in all attributes gives us characteristic rules. We can then use a single rule induction system as a unified generator of discrimination and characteristic rules.

4 Experimental Results and Discussion

The method proposed in the previous section was implemented as DISCAS version 2 software using lisp. A Pentium II 448MHz PC was used in all experiments and the database was stored in memory. The following were the three input parameters used in the DISCAS software.

1. *Minsup*: the minimum support employed in association rule mining.
2. $thres_i$: a parameter to prune the link expansion introduced in Sects. 3.2-3.3.
3. $min\text{-}BSS_i$: a link written out as a rule candidate when one of its BSS_i values along the link exceeds this parameter.

Characteristic rules are derived from a House voting-record dataset with 17 attributes and 435 cases [13] to estimate the performance of DISCAS. Table 2 shows the number of nodes, the elapsed time to generate the lattice, and the number of resulting rules changing the first two parameters. The values of $thres_i$ are set equal for all attributes, and the values for $min\text{-}BSS_i$ are set to 10% of the SS_i for the entire dataset. All candidate links are adopted as rules. To avoid the confusion created by the effects of various m_i values in the attributes, pruning was done assuming that all m_i were equal to 2 in (4).

The row with *thres* •= •0.0 in Table 2 shows the results without pruning by the *thres* values. Results in the other rows indicate that the application of pruning has been very effective in reducing the lattice size and the computation time, which are roughly proportional if the lattice size is not large. When *thres* or *minsup* are in a low value range, the number of rules does not always increase even if they take lower values, because a link with few instances cannot give enough *BSS* to exceed $min\text{-}BSS_i$.

Next, we inspect the results in the column for which *minsup* = 0.05. Figure 4 shows the number of nodes at each generation of the lattice changing *thres*, where we can see a typical profile of the lattice size constructed from table data. Remarkable pruning effects are observed when the number of items in an itemset reaches four.

Pruning should not diminish strong rules. It is interesting to investigate the distribution of the *BSS* values of the rules changing *thres*. The maximum value in BSS_i's along a link, called $maxBSS_i$, is examined. Table 3 shows the number of rules classified by $maxBSS_i$ and by *thres* at *minsup* = 0.05. The headline shows the minimum value of $maxBSS_i$ for each column, where N is 435.

The number of rules with pruning is not changed from the number without pruning (*thres* = 0.0), as shown in the upper right region partitioned by the solid line. There are 27 strong interactions that do not change even at *thres* = 0.05. The pruning condition denotes that a substantial decrease in rule counts may be observed at the lower left region of the broken line, where $maxBSS_i$ is less than N•••*th r e s*. However, there is a large number of pruned rules in all the cells of the leftmost column. Either way, we can expect that strong rules will not be affected by pruning, even if we use high *thres* values.

When our aim is to find characteristic rules, the strength of a rule should be judge by the sum of BSS_i values along a link. When we used this criterion for rule selection more than 152 rules were never affected, even at *thres* = 0.05.

Table 2. Number of nodes, time period and number of rules changing *minsup* and *thres*, where *time* is the elapsed time in seconds. Note that — indicates that computation was not accomplished due to memory limitations.

thres		*minsup*					
		0.010	0.025	0.050	0.100	0.150	0.200
0.00	nodes		882714	337216	92747	31081	13933
	time	—	224313	39695	2375	626	154
	rules		808	642	350	218	143
0.01	nodes	348196	136244	50986	14200	4831	2214
	time	23514	2692	501	98	33	17
	rules	731	731	628	350	218	143
0.02	nodes	98929	41834	16481	4998	2040	900
	time	1061	313	101	31	14	7
	rules	678	678	598	349	218	143
0.03	nodes	46199	21098	8921	2895	1306	589
	time	301	114	48	18	9	5
	rules	614	614	554	340	215	142
0.04	nodes	25132	12460	5515	1911	914	442
	time	137	61	28	11	7	3
	rules	604	604	547	340	215	142
0.05	nodes	15643	8148	3853	1429	728	355
	time	73	40	20	9	5	3
	rules	560	560	510	332	214	141

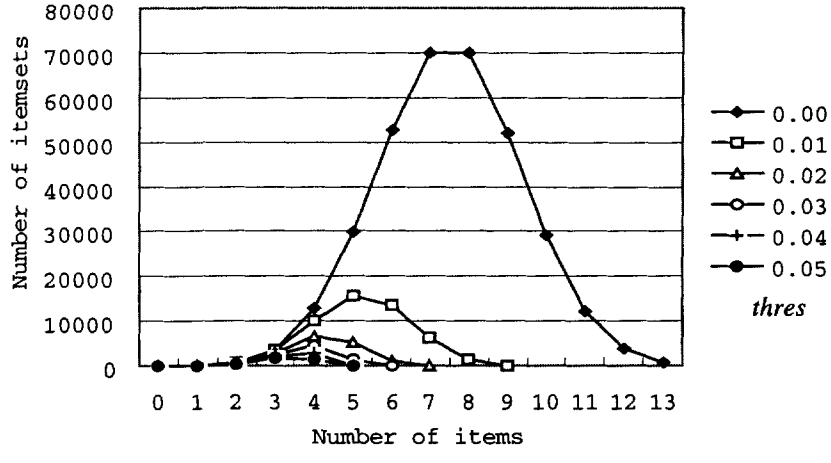


Fig. 4: Number of itemsets for each level of lattice; variable *thres*, *minsup* fixed at 0.05.

Table 3. Number of rules classified by $maxBSS_i$ and *thres* at *minsup* = 0.05

thres	$maxBSS_i$						
	0.03·*N*	0.04·*N*	0.05·*N*	0.06·*N*	0.07·*N*	0.08·*N*	0.09·*N*
0.00	264	90	33	12	7	5	3
0.01	253	90	33	12	7	5	3
0.02	238	89	33	12	7	5	3
0.03	217	86	32	12	7	5	3
0.04	213	86	32	12	7	5	3
0.05	198	81	31	12	7	5	3

5 Concluding Remarks

A pruning methodology based on the *SS* criterion has provided an effective framework for rule induction. The efficiency of pruning is very useful in table data, which has been hard to handle because of the combinatorial explosion in the number of nodes. This method is also applicable to market basket analysis. Low interactions among most items are expected to lead to effective pruning in lattice generation. It will be useful if the cost of database access is higher than that of the item counting operations.

The dynamic output of rule links also enables the detection of interactions when the expansion of a lattice to higher levels is impossible. It can be used in real time applications like OLAP and a text mining system for the WWW.

Developed software can easily control the appearance of attributes in the LHS and the RHS of a rule. Fine-tuning of parameters based on field expertise enables fast and effective mining that can analyze not only demographic data but also transactions' data. Analysis of the combined dataset of these two styles will be necessary in future scientific discovery, such as pattern extraction from clinical histories and the detection of specific effects from laboratory notebooks. The DISCAS software is publicly available to academic users upon request to the author.

As the sum of squares criterion constitutes one of the core analysis criterion in the statistics of continuous variables, the proposed method is expected to lead to a unified and seamless architecture in data analysis when the detection of local interactions is important.

References

[1] Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. Proc. ACM SIGMOD (1993) 207-216

[2] Ali, K., Manganaris, S., Srikant, R.: Partial Classification using Association Rules. Proc. KDD-97 (1997) 115-118

[3] Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. Proc. KDD-98 (1998) 80-86

[4] Meretakis, D., Wüthrich, B.: Classification as Mining and Use of Labeled Itemsets. Proc. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (1999)
[5] Silverstein, C., Brin, S., Motwani, R.: Beyond Market Baskets: Generalizing Association Rules to Dependence Rules. Data Mining and Knowledge Discovery, 2 (1998) 39-68
[6] Okada, T.: Finding Discrimination Rules using the Cascade Model. J. Jpn. Soc. Artificial Intelligence, 15 (2000) in press
[7] Okada, T.: Rule Induction in Cascade Model based on Sum of Squares Decomposition. Principles of Data Mining and Knowledge Discovery (Proc. PKDD'99), 468-475, Lecture Notes in Artificial Intelligence 1704, Springer-Verlag (1999).
[8] Okada, T.: Sum of Squares Decomposition for Categorical Data. Kwansei Gakuin Studies in Computer Science 14 (1999) 1-6. http://www.media.kwansei.ac.jp/home/kiyou/kiyou99/kiyou99-e.html
[9] Gini, C.W.: Variability and Mutability, contribution to the study of statistical distributions and relations, *Studi Economico-Giuridici della R. Universita de Cagliari* (1912). Reviewed in Light, R.J., Margolin, B.H.: An Analysis of Variance for Categorical Data. J. Amer. Stat. Assoc. 66 (1971) 534-544
[10] Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. Proc. VLDB (1994) 487-499
[11] Toivonen, H.: Sampling Large Databases for Finding Association Rules. Proc. VLDB (1996) 134-145
[12] Brin, S., Motwani, R., Ullman J. D., Tsur, S.: Dynamic Itemset Counting and Implication Rules for Market Basket Data. Proc. ACM SIGMOD (1997) 255-264
[13] Mertz, C. J., Murphy, P. M.: UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, Dept. of Information and Computer Sci. (1996)

Appendix

We give a proof for the upper bound of BSS_i shown in (3).

$$BSS_i \le \frac{m_i \cdot n^{\mathrm{L}}}{2}\left(1-p_0^{\mathrm{U}}(a_0)\right)^2 \quad , \tag{3}$$

where U and L denote the upper and the lower nodes of a link, along which an item $[x_0: a_0]$ is added, m_i is the number of attribute values for x_i, n^{L} is the number of cases on L, and $p_i(a)$ is the probability of attribute x_i having the value a. The expressions of BSS_i and n^{L} are given by,

$$BSS_i = \frac{n^{\mathrm{L}}}{2}\sum_a \left(p_i^{\mathrm{L}}(a)-p_i^{\mathrm{U}}(a)\right)^2 \tag{5}$$

$$n^{\mathrm{L}} = n^{\mathrm{U}} \cdot p_0^{\mathrm{U}}(a_0) \tag{6}$$

Then the following inequalities hold.

$$\begin{aligned} 0 \le n^{\mathrm{L}} \cdot p_i^{\mathrm{L}}(a) \le n^{\mathrm{U}} \cdot p_i^{\mathrm{U}}(a) \quad , \\ 0 \le n^{\mathrm{L}} \cdot \left(1-p_i^{\mathrm{L}}(a)\right) \le n^{\mathrm{U}} \cdot \left(1-p_i^{\mathrm{U}}(a)\right) \quad . \end{aligned} \tag{7}$$

The bounds of $p_i^{\mathrm{L}}(a)$ are expressed by,

$$p_i^{\mathrm{L}}(a)\begin{cases}\geq 0 & \text{if } p_i^{\mathrm{U}}(a)\leq 1-p_0^{\mathrm{U}}(a_0) \quad , \\ \geq 1-\dfrac{1-p_i^{\mathrm{U}}(a)}{p_0^{\mathrm{U}}(a_0)} & \text{if } p_i^{\mathrm{U}}(a)>1-p_0^{\mathrm{U}}(a_0) \quad , \end{cases} \tag{8a}$$

$$p_i^{\mathrm{L}}(a)\begin{cases}\leq 1 & \text{if } p_i^{\mathrm{U}}(a)\geq p_0^{\mathrm{U}}(a_0) \quad , \\ \leq \dfrac{p_i^{\mathrm{U}}(a)}{p_0^{\mathrm{U}}(a_0)} & \text{if } p_i^{\mathrm{U}}(a)< p_0^{\mathrm{U}}(a_0) \quad . \end{cases} \tag{8b}$$

Here, we regard (5) as a quadratic form of $\{p_i^{\mathrm{L}}(a)\}$. Since it takes the minimum at $\{p_i^{\mathrm{U}}(a)\}$ and its region is constrained by (8) on a hyperplane defined by $\sum_a p_i^{\mathrm{L}}(a)=1.0$, BSS_i takes the maximum value at some boundary point. Here, we use a notation $q(a)$ to denote the value of $p_i^{\mathrm{L}}(a)$ at the maximum point of BSS_i. First, let us consider the case that $q(a)$ is at the higher boundary in the region, where $q(a)$ - $p_i^{\mathrm{U}}(a)$ is positive.

if $p_i^{\mathrm{U}}(a) < p_0^{\mathrm{U}}(a_0)$ then

$$q(a)-p_i^{\mathrm{U}}(a)\leq \frac{p_i^{\mathrm{U}}(a)}{p_0^{\mathrm{U}}(a_0)}-p_i^{\mathrm{U}}(a)=\frac{p_i^{\mathrm{U}}(a)}{p_0^{\mathrm{U}}(a_0)}\left(1-p_0^{\mathrm{U}}(a_0)\right)\leq 1-p_0^{\mathrm{U}}(a_0) \quad , \tag{9}$$

if $p_i^{\mathrm{U}}(a)\geq p_0^{\mathrm{U}}(a_0)$ then

$$q(a)-p_i^{\mathrm{U}}(a)\leq 1-p_i^{\mathrm{U}}(a)\leq 1-p_0^{\mathrm{U}}(a_0) \quad .$$

On the other hand, if $q(a)$ is at the lower boundary, the following inequalities hold.

if $p_i^{\mathrm{U}}(a)>1-p_0^{\mathrm{U}}(a_0)$ then

$$\begin{aligned} p_i^{\mathrm{U}}(a)-q(a) &\leq p_i^{\mathrm{U}}(a)-\left(1-\frac{1-p_i^{\mathrm{U}}(a)}{p_0^{\mathrm{U}}(a_0)}\right)=\frac{1-p_i^{\mathrm{U}}(a)}{p_0^{\mathrm{U}}(a_0)}\left(1-p_0^{\mathrm{U}}(a_0)\right) \\ &\leq 1-p_0^{\mathrm{U}}(a_0) \quad , \end{aligned} \tag{10}$$

if $p_i^{\mathrm{U}}(a)\leq 1-p_0^{\mathrm{U}}(a_0)$ then

$$p_i^{\mathrm{U}}(a)-q(a)\leq 1-p_0^{\mathrm{U}}(a_0)-q(a)\leq 1-p_0^{\mathrm{U}}(a_0) \quad .$$

Then, we obtain the following inequality,

$$\left(q(a)-p_i^{\mathrm{U}}(a)\right)^2 \leq \left(1-p_0^{\mathrm{U}}(a_0)\right)^2 \quad . \tag{11}$$

As (11) holds for any value a of an attribute x_i, introduction of (11) into (5) gives the proof of (3).

The author anticipates that (3) will hold for $m_i = 2$. We have found no violations to this stricter bound during extensive numerical checks. The proof of this inequality is expected.

Extracting Predictors of Corporate Bankruptcy: Empirical Study on Data Mining Methods

Cindy Yoshiko Shirata[1] and Takao Terano[2]

[1] Tsukuba College of Technology Japan,
4-12 Kasuga Ibaraki-Pref. 305-0831 Japan
cindy@cs.k.tsukuba-tech.ac.jp
[2] University of Tsukuba, 3-29-1 Otsuka, Bunkyo-ku
Tokyo 112- , Japan
terano@gssm.otsuka.tsukuba.ac.jp

Abstract. We presents some empirical results of a study regarding financial ratios as predictors of Japanese corporate bankruptcy based on a large sample of bankrupt and non-bankrupt firms' financial data. In this study, variable as predictors of bankruptcy had been selected by three AI-based data mining techniques and two conventional statistical methods, Logit analysis and Stepwise. After the selection of a set of variables for every method, discriminant power of each set was compared to verify the most suitable data mining technique to select financial variables. Finally, the study concludes that a set of variables selected by Logit analysis (with logit model) indicated the best discriminant power, more than 87% accuracy.

1 Introduction

We attempted to obtain sets of the financial ratios as predictors using some different kinds of AI-based data mining methods with a large sample of financial data. It is also selected another sets of financial ratios with conventional statistical techniques. Each method selects one set of financial variables. After the selection of a set of financial ratios for every data mining method, we compare the discriminant power of all sets with four different multivariate discriminant analysis, linear, quadratic, normal-kernel method and logit analysis. This procedure identifies which set of financial variables can best predict corporate bankruptcy in Japan and the most suitable data mining method to analyze financial variables.

2 Methodology

We analyze financial variables with three different AI-based data mining techniques. One is C4.5 that is a quite popular data mining tool [1]. The other one is SIBILE

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 204-207, 2000.
© Springer-Verlag Berlin Heidelberg 2000

which is using interactive Genetic Algorithms and inductive learning techniques [2]. Final one is Classification and Regression Trees (CART) [3] providing from S-plus program. These three techniques can treat not only quantitative variables but also qualitative variables. Therefore, two qualitative variables, Capital size and Industry category will be added on analysis.

This study also tries to select the financial variables with two conventional statistical techniques, Stepwise of SAS program (SAS ver.6.12) and Logit analysis of S-plus program (S-plus ver3.1).

3 Variables and Sample Design

The original sixty-six financial variables were chosen on the basis of (1) popularity in literature, (2) usage by the Japan Development Bank, (3) usage by Teikoku Data Bank[1] of its Cosmos 1 credit database, and (4) the author's initiated hypothesis. The variables can be classified into the following categories (1) popularity in the literature (X1-X8), (2) growth (X9-X12), (3) capital efficiency (X13-X17), (4) profitability (X18-X27), (5) activity (X28-X42), (6) productivity (X43-X47), (7) liquidity (X48-X53) and (8) coverage and other earnings relative to leverage measures (X54-X66). Two qualitative variables, Capital size and Industry category are added as variables.

The samples include all bankrupt firms obtained from Teikoku Data Bank Cosmos1 Database. The data set for this study is 686 bankrupt firms and 300 non-bankrupt firms. The 300 non-bankrupt firms were extracted from 107,034 non-bankrupt firms by systematic sampling method. All the bankrupt firms had failed between 1986 to 1996 in Japan.

4 Data Cleaning

The most important data cleaning procedure is de-duplication of records. All data using in this study is checked completely that there is no duplication. However, here is some other very important data cleaning procedure when we treat financial variables. First, if there is a missing value on data, some programs do not work well or sometimes cause error. Therefore, it is necessary to delete a whole data line that has a missing value. We delete such data in advance with cautiously. Second, it must confirm the relationship between corporate behavior and tend of financial ratios. There is a basic rule of financial analysis when we evaluate firms. This rule came from traditional accounting ethics. To confirm whether the distribution of each variable follows accounting ethics, univariate approach has been taken. The result of the univariate analysis of all variables, twelve variables showed the completely opposite tend from a

[1] About 1,000,000,000 financial statements of 250,000 Japanese firms (all industry and size) are stored in Teikoku Data Bank's Cosmos1Data Base. The company also has the Bankruptcy Data File which contains the name of failed firms, the date of filing and reason for bankruptcy.

basic financial analysis rule coming from accounting ethics. We decided to delete these variables which were indicated the opposite tend from ethics. Because they may mislead the results of analysis toward wrong direction.

5 Experimental Results

Some methods selected too many variables. Therefore, we choose suitable number of variables which can represent the result of each method. C4.5 assigned fifty variables on decision tree. We decided to choose variables indicating by the sixth node, thirteen variables. The important factor to decide what variables we choose is larger likelihood ratio of variables. The likelihood ratio of variables would go down drastically from the seventh node. And there was also an interesting result that only C4.5 selected a qualitative variable, Capital size, on the seventh node. However, no other technique selected qualitative variables. After consideration whether this variable should be included, we decided not to add this variables for the next analysis. We also concluded here that Size or Industry variable does not influence bankruptcy phenomenon more than financial variables.

CART chosen twenty-four variables. However, the variables at over 100 nodes seem not to be contributed well for the model, and it was also confirmed that over 100 nodes variables indicated smaller likelihood ratio. Hence, we decided to choose nineteen variables selected at less than 100 nodes on CART.

Logit analysis selected twenty-three variables. We choose nineteen higher likelihood ratio variables which are also indicating higher CP value. Table 1 presents the results of comparison of discriminant power of each set of variables.

Table 1. Comparison of Discriminat Power with Misclassification Rate

Method	No. of variables	Normal Kernel	Linear	Quaratic	Logit	Best Model	prob
C4.5	13	0.219	0.175	0.332	0.152	logit	0.152
SIBIL	10	0.184	0.184	0.194	0.175	logit	0.175
CART	19	0.204	0.160	0.195	0.177	linear	0.160
Logit	19	0.205	0.140	0.219	0.128	logit	0.128
Stepdisc	17	0.230	0.181	0.275	0.166	logit	0.166

Based on the result of our analysis, a set of variables selected by Logit analysis (with logit model) indicated the best discriminant power, more than 87% accuracy. The same set of variables with linear model showed the second discriminat power, more than 86%. Therefore, we can conclude that the Logit analysis is the best data mining method to extract predictors of Japanese corporate bankruptcy from financial variables.

Another notable point here is that sets of variables selected by C4.5 and CART were also indicating higher discriminant power (around 85%) following a set of variables selected by Logit analysis. Since we got an adequate result from these AI-based data mining techniques, it convinced us that these techniques were available for variable selection concerning financial problems.

Furthermore, all sets prove that linear model or logit model is a suitable model to predict bankruptcy with financial variables. In contrast, Normal Kernel (non-parametric model) and quadratic model are not suitable for bankruptcy prediction model.

6 Conclusion

Data mining techniques usually analyze a large sample of data like ten thousand or more. In contrast, this study treated only one thousand data. However, previous studies developing a bankruptcy prediction model or treating financial problems analyzed only fifty to hundred sample data. Therefore, the results of these studies were not generalizable, due to limited size of their samples. In contrast, this study treated the largest sample of data in this kind of study that had been done ever, and we believe that our results are more reliable than ever.

There are some interesting results on this study. All AI-based techniques selected X2 for the best predictor that can discriminate bankrupt firms significantly. In contrast, conventional statistical methods did not select X2 at all. Altman mentioned in his study that X2 was unquestionably the most important variable [4]. This result impressed us that variable selection by AI-based techniques is more trustworthy than variable selection by conventional statistical method.

In spite of each combination of selected variables are different, all model indicate an adequate discriminant power. That teaches us that financial variables have discriminant power originally whether the firm is in critical condition. If you would like to get a few percent higher prediction, it is worth selecting variables by conventional methods like Logit analysis. However, if you would like to have more stable results, AI-based techniques like C4.5 or CART can provide a sound set of variables with reasonable discriminant power.

References

1. Quinlan, J.R.: Introduction of decision trees. Machine Learning 1 (1986) 81-106.
2. Terano, T., Ishino Y.,: Interactive Generic Algorithm Based Feature Selection and its Application to Marketing Data Analysis. Feature Extraction, Construction and Selection: A Data Mining Perspective, Massachusetts (1998) 393-406.
3. Breimann, L., J.H. Frieman, R.A. Olshen, and C.J. Stone: Classification and Regression Trees. Chapman & Hall, London (1984).
4. Altman, E., R. G. Haldeman and P. Narayanan: ZETA Analysis: A new model to indentify bankruptcy risk of corporations, Journal of Banking and Finance Vol.1, June (1977) 35.

Evaluating Hypothesis-Driven Exception-Rule Discovery with Medical Data Sets

Einoshin Suzuki[1] and Shusaku Tsumoto[2]

[1] Division of Electrical and Computer Engineering, Faculty of Engineering, Yokohama National University
suzuki@dnj.ynu.ac.jp
[2] Department of Medical Informatics, Shimane Medical University, School of Medicine
tsumoto@computer.org

Abstract. This paper presents a validation, with two common medical data sets, of exception-rule discovery based on a hypothesis-driven approach. The analysis confirmed the effectiveness of the approach in discovering valid, novel and surprising knowledge.

1 Introduction

In rule discovery, a discovered rule can be classified as either a common sense rule, which holds true for many examples, or an exception rule, which represents a different regularity from a common sense rule [1–4] ! %An exception rule often exhibits unexpectedness and usefulness since it differs from a common sense rule which is often well-known. A hypothesis-driven method obtains a set of pairs of an exception rule and a common sense rule [1–4]. This method is supposed to discover unexpected rules since it is independent of user-supplied domain knowledge. In this paper, we validate this method using two data sets [5].

2 Problem Description

Let an atom represent an event which is either a single value assignment to a discrete attribute or a single range assignment to a continuous attribute. We define a conjunction rule as a production rule of which premise is represented by a conjunction of atoms and conclusion is a single atom.

Suzuki, one of the authors, considered a problem of finding a set of rule pairs [1–4]. Here, a rule pair $r(x, x', Y_\mu, Z_\nu)$ is defined as a pair of two conjunction rules, which are a common sense rule $Y_\mu \rightarrow x$ and an exception rule $Y_\mu \wedge Z_\nu \rightarrow x'$.

$$r(x, x', Y_\mu, Z_\nu) \equiv \{Y_\mu \rightarrow x,\ Y_\mu \wedge Z_\nu \rightarrow x'\}$$

where x and x' are a single atom with the same attribute but different values. Each premise of rules represents a conjunction of atoms $Y_\mu \equiv y_1 \wedge y_2 \wedge \cdots \wedge y_\mu$, $Z_\nu \equiv z_1 \wedge z_2 \wedge \cdots \wedge z_\nu$.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 208-211, 2000.
© Springer-Verlag Berlin Heidelberg 2000

The method employed in this paper outputs rule pairs which satisfy

$$\widehat{\Pr}(Y_\mu) \geq \theta_1^{\mathrm{S}},\ \widehat{\Pr}(x|Y_\mu) \geq \mathrm{MAX}(\theta_1^{\mathrm{F}}, \widehat{\Pr}(x)),\ \widehat{\Pr}(Y_\mu, Z_\nu) \geq \theta_2^{\mathrm{S}},$$
$$\widehat{\Pr}(x'|Y_\mu, Z_\nu) \geq \mathrm{MAX}(\theta_2^{\mathrm{F}}, \widehat{\Pr}(x')),\ \widehat{\Pr}(x'|Z_\nu) \leq \mathrm{MIN}(\theta_2^{\mathrm{I}}, \widehat{\Pr}(x'))$$

where $\widehat{\Pr}(x)$ represents the ratio of an event x in the data set, and each of $\theta_1^{\mathrm{S}}, \theta_1^{\mathrm{F}}, \theta_2^{\mathrm{S}}, \theta_2^{\mathrm{F}}, \theta_2^{\mathrm{I}}$ is a user-supplied threshold.

3 Application to the Meningitis Data Set

3.1 Conditions of the Application

The updated version of the meningitis data set [5] consists of 140 patients each of whom is described with 38 attributes. Here, a length of a premise in a rule pair is limited to one, i.e. $\mu = \nu = 1$, in the application. The other parameters were settled as $\theta_1^{\mathrm{S}} = 0.2$, $\theta_1^{\mathrm{F}} = 0.75$, $\theta_2^{\mathrm{S}} = 5/140$, $\theta_2^{\mathrm{F}} = 0.8$, $\theta_2^{\mathrm{I}} = 0.4$.

Tsumoto, a domain expert, evaluated each discovered rule-pair from the viewpoint of validness, novelty, unexpectedness, and usefulness. For each index of a rule pair, he attributed an integer score ranging from one to five. A zero score was attributed if he judged necessary.

3.2 Average Results and Analysis

Table 1 shows results of the experiment described in the previous section. From the table, we see that the method outputted 169 rule pairs, and their average performance is 2.9, 2.0, 2.0, and 2.7 for validness, novelty, unexpectedness, and usefulness. Note that it is relatively easy to discovery valid or useful rule pairs than novel or unexpected rule pairs. We inspected these rule pairs by grouping them with respect to the attribute in the conclusion, and found that these attributes can be classified into four categories. The first category represents attributes with the lowest scores, and includes CULTURE, C_COURSE, and RISK. We consider that attributes in this category cannot be explained with this data set, and investigation on them requires further information on other attributes. The second category represents attributes with higher scores for validness and usefulness, and includes FOCAL, LOC_DAT, and Diag2. We consider that attributes in this category can be explained with this data set, and has been well investigated probably due to their importance in this domain. We regard them as one of important targets in discovery although one will often rediscover conventional knowledge. The third category represents attributes with approximately equivalent scores, and includes CT_FIND, EEG_FOCUS, and Course (G). We consider that attributes in this category can be explained with this data set, and has not been investigated well in spite of their importance in this domain. We regard them as one of the most important targets in discovery. The fourth category represents attributes with higher scores for novelty and unexpectedness, and includes CULT_FIND, KERNIG, and SEX. We consider that attributes in

Table 1. Average performance of the proposed method with respect to attributes in the conclusion. The column "#" represents the number of discovered rule pairs.

attribute	#	validness	novelty	unexpectedness	usefulness
all	169	2.9	2.0	2.0	2.7
CULT_FIND	4	3.3	4.0	4.0	3.5
CT_FIND	36	3.3	3.0	3.0	3.2
EEG_FOCUS	11	3.0	2.9	2.9	3.3
FOCAL	18	3.1	2.2	2.7	3.0
KERNIG	4	2.0	3.0	3.0	2.0
SEX	1	2.0	3.0	3.0	2.0
LOC_DAT	11	2.5	1.8	1.8	2.5
Diag2	72	3.0	1.1	1.1	2.6
Course (G)	8	1.8	2.0	2.0	1.8
CULTURE	2	1.0	1.0	1.0	1.0
C_COURSE	1	1.0	1.0	1.0	1.0
RISK	1	1.0	1.0	1.0	1.0

this category can be explained with this data set, but has been somewhat ignored. We consider that investigating these attributes using discovered rule sets can lead to interesting discoveries which might reveal unknown mechanisms in this domain in spite of their apparent low importance.

3.3 Examples of Discovered Rule Pairs

We have also pursued a best-case analysis, and found it much more promising as expected. For instance, the following rule pair has a four-rank score for every index.

```
83=<CSF_PRO=<121           ->CULT_FIND=F
83=<CSF_PRO=<121, FOCAL=+ ->CULT_FIND=T
```

This rule pair has the following statistics: $\widehat{\Pr}(Y_\mu) = 0.257$, $\widehat{\Pr}(x|Y_\mu) = 0.778$, $\widehat{\Pr}(Y_\mu, Z_\nu) = 0.035$, $\widehat{\Pr}(x'|Y_\mu, Z_\nu) = 1.000$, $\widehat{\Pr}(x'|Z_\nu) = 0.285$. In other words, among 140 patients, 36 patients had `83=<CSF_PRO=<121`, and 78 % of them were also `CULT_FIND=F`. However, five patients who were `FOCAL=+` in addition to `83=<CSF_PRO=<121` were actually all `CULT_FIND=T`. This exception rule is interesting since only 28.5 % of patients who were `FOCAL=+` were `CULT_FIND=T`.

Tsumoto also found several rule pairs concerning EEG_FOCUS, Diag2, FOCAL and CT_FIND very interesting. In a paper [5] comparing eleven KDD methods with respect to this data set, he states this method as "structure of rule pairs is very appealing to medical experts". He also admits that this method discovered the most interesting results among eleven methods.

4 Application to the Bacterial Test Data Set

The bacterial test data set represents a version preprocessed by Suzuki. It consists of 20,919 patients each of whom is admitted in a hospital and is described with 135 attributes. In the application, the parameters were settled as $\mu + \nu \leq 2$, $\theta_1^{\mathrm{S}} = 0.01$, $\theta_1^{\mathrm{F}} = 0.7$, $\theta_2^{\mathrm{S}} = 5/20919$, $\theta_2^{\mathrm{F}} = 0.8$, $\theta_2^{\mathrm{I}} = 0.5$. We resolved two tasks given with this data set, and thus restricted the attributes in the conclusion to the existence of bacteria and effectiveness of various antibiotics.

The bacterial test data set has been provided by a hospital in Japan, and related domains are so diverse that complete domain-knowledge has not been established. Therefore, unlike the meningitis data set, it it difficult to score the results. Here, we show only one of the rule pairs that Tsumoto felt the most interesting.

```
sex=M, LCMs=sensitive                    ->PCG=registant
sex=M, LCMs=sensitive, ward=surgery ->PCG=sensitive
```

where "LCMs" and "PCG" are both antibiotics. The common sense rule holds true 52.8 % for 598 patients, while the exception rule holds true for 34 patients among 46. However, among the patients who were `ward=surgery`, only 1.6 % were `PCG=sensitive`. This rule pair is especially important since it can reveal mechanisms related to antibiotics as well as discover anomalies in a ward.

5 Conclusions

In this paper, a hypothesis-driven discovery method for exception rules was applied to two medical data sets, and the results were evaluated by a domain expert. An average-case analysis for the meningitis data set showed that this approach is highly effective. A best-case analysis for the bacterial test data set also led to discovery of highly interesting knowledge.

References

1. E. Suzuki and M. Shimura : Exceptional Knowledge Discovery in Databases Based on Information Theory, *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD-96)*, AAAI Press, Menlo Park, Calif., pp. 275–278 (1996).
2. E. Suzuki: "Autonomous Discovery of Reliable Exception Rules", *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining (KDD-97)*, AAAI Press, Menlo Park, Calif., pp. 259–262 (1997).
3. E. Suzuki and Y. Kodratoff: "Discovery of Surprising Exception Rules based on Intensity of Implication", *Principles of Data Mining and Knowledge Discovery (PKDD'98), LNAI 1510*, Springer, Berlin, pp. 10–18 (1998).
4. E. Suzuki: "Scheduled Discovery of Exception Rules", *Discovery Science (DS'99), LNAI 1721*, Springer, Berlin, pp. 184–195 (1999).
5. S. Tsumoto *et al.*: "Comparison of Data Mining Methods using Common Medical Datasets", *ISM Symposium: Data Mining and Knowledge Discovery in Data Science*, The Inst. of Statistical Math., Tokyo, pp. 63–72 (1999).

Discovering Protein Functional Models Using Inductive Logic Programming

Takashi Ishikawa[1], Masayuki Numao[2], and Takao Terano[3]

[1] Dept. of Information and Computer Eng., Kisarazu National College of Technology, Chiba 292-0041, Japan,
takashi@j.kisarazu.ac.jp,
http://www.kisarazu.ac.jp/~jisikawa/
[2] Dept. of Computer Sci., Faculty of Eng., Tokyo Institute of Technology, Tokyo 152-8552, Japan,
numao@cs.titech.ac.jp,
http://www.nm.cs.titech.ac.jp/numao/
[3] Graduate School of Systems Management, The University of Tsukuba, Tokyo 112-0012, Japan,
terano@gssm.otsuka.tsukuba.ac.jp,
http://www.gssm.otsuka.tsukuba.ac.jp/staff/~terano/

Abstract. The paper describes a method for machine discovery of *protein functional models* from protein databases using *Inductive Logic Programming* based on top-down search for *relative least general generalization*. The method discovers effectively protein function models that explain the relationship between functions of proteins and their amino acid sequences described in protein databases. The method succeeds in discovering protein functional models for forty membrane proteins, which coincide with conjectured models in literature of molecular biology.

1 Introduction

Inductive Logic Programming (ILP) [9] has succeeded in applications to molecular biology including secondary structure prediction of protein [8] and other problems [3]. However, ILP has not been applied to the central problem that is to explain the relationship between protein functions and their amino acid sequences. The paper aims at solving the problem of *protein function prediction* by discovering *protein functional models* [5] using ILP.

Traditional methods for protein function prediction use *homology search* and *sequence motif* [1]. Homology search uses global similarities of amino acid sequences to find protein of similar functions. On the other hand, sequence motifs are local patterns of amino acid sequences that are unique to certain functions of proteins and are stored in the database for some protein functions [2]. Protein function prediction by sequence motifs is based on matching the target amino acid sequence with sequence motifs in the database. These methods use global or local similarities among amino acid sequences to find protein of similar functions. Therefore these methods are limited to proteins with almost same amino

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 212–215, 2000.
© Springer-Verlag Berlin Heidelberg 2000

acid sequences in global or local regions from their fundamental principles. An another method for protein function prediction uses 3D structures of proteins, but 3D structures are difficult to predict from amino acid sequences.

2 Protein Functional Models

The approach of the paper to protein function prediction is based on the assumption that combinations of functional sites, which are sequence patterns of amino acid sequences, characterize a protein function. Functional sites are associated with secondary structures of proteins in order to specify their position in the 3D structures. In our protein functional models, functional sites are represented by strings of characters that code amino acids and combination of functional sites are used to discriminate narrow functional difference. Furthermore, using protein functional models allows us to predict protein functions from only amino acid sequences instead of requiring geometrical information representing 3D structures.

A protein functional model is represented by the following clause in a logical.

$$protein(ID, FUNCTION) \leftarrow subseq(ID, PATTERN, POS/STD), ...$$

The head of the clause is a literal representing that the protein ID has $FUNCTION$ using predicate *protein*. The body of the clause is a conjunction of literal representing that the protein ID has subsequence pattern $PATTERN$ at POS/STD using predicate *subseq*. Here POS stands for *secondary structure position* of the pattern and STD stands for standard protein's ID for which POS is given. The secondary structure position is determined by finding the most similar subsequence in the amino acid sequence of the protein STD in the corresponding secondary structure.

3 A Method for Discovery

The approach of the paper to protein function prediction employs inductive logic programming to discover protein functional models. Inductive logic programming is a machine learning technique suitable for generating hypotheses represented by first order predicate logic allowing to describe elements of objects and relation among elements like protein functional models. Unfortunately, traditional inductive logic programming systems such as Progol [10] and FOIL [11] are difficult to apply to discovering of protein functional models because of their restriction for hypothesis language.

We have developed an ILP method [6] that satisfies the requirements above, which integrates a top-down method and a bottom-up method of inductive logic programming. The method is based on the top-down search utilizing an information theoretic heuristic used in FOIL [11] and generate literals in the hypothesis clause using *relative least general generalization* (*rlgg*) used in GOLEM [7] in a bottom-up manner. The information theoretic heuristic makes the method efficient instead using mode declaration. The use of *rlgg* enables the method to

generate hypotheses involving literals with function terms in order to describe sub sequence patterns as a list of characters.

In the top-down search for hypothesis clauses, the method generates *selectively* literals satisfying the following conditions in order to generate clauses with less redundant literals.

(a) having common variable(s) with existing literals in the hypothesis clause
(b) being *lgg* of two ground unit clauses in the background knowledge
(c) giving information gain when the literal is added to the body of the clause

4 Experiment

We have conducted learning experiments in which the discovered results are compared to the known functional models to evaluate the effectiveness of the proposed method. The materials are forty membrane proteins in the protein database SWISS-PROT [2] listed below, for which protein functional models are known in the literature of molecular biology.

Input data and the discovering program are described by MacProlog32 and the computation is performed on Power Macintosh 8100/100AV. The sum of positives and negatives is forty and the number of background unit clauses is about 16000.

Bacteriorhodopsin is a protein that exist in cell membrane of a special bacteria and has protein function of *proton pump* which transports proton (i.e., hydrogen ion) using photo energy. The functional sites of *bacteriorhodopsin* are considered to be three amino acids D, K, D in the amino acid sequence.

Figure 1 shows the correspondence between the experimental results and the known functional sites in the amino acid sequence of *bacteriorhodopsin* [4]. The numbers above amino acid in Figure 1 indicate the number of trans membrane domain and symbols '+' and '-' denote specific amino acids in the discovered functional sites and any amino acids respectively. The correspondence indicates that the method re-discovered all the functional sites of *bacteriorhodopsin.*

5 Conclusion

The paper described a method to discover *protein functional models* from protein databases using Inductive Logic Programming based on top-down search for relative least general generalization. Protein function models explain the relationship between protein functions and their amino acid sequences described in protein databases. The method succeeded in discovering protein functional models for forty membrane proteins, which coincide with conjectured functional models in the literature of molecular biology.

```
                          11111111 1111111111
MDPIALTAAV GADLLGDGRP ETLWLGIGTL LMLIGTFYFI

1111           22222 2222222222 2222222222
VKGWGVTDKE AREYYSITIL VPGIASAAYL SMFFGIGLTE

2              3*333 3333333*33 333      4
VQVGSEMLDI YYARYADWLF TTPLLLLDLA LLAKVDRVSI
              ++--+     -+++++++

4444444444 444444444          5 5555555555
GTLVGVDALM IVTGLVGALS HTPLARYTWW LFSTICMIVV

5555555555                66666 6666666666
LYFLATSLRA AAKERGPEVA STFNTLTALV LVLWTAYPIL
                                        ++--

66666       77777777 7777777*77 777777
WIIGTEGAGV VGLGIETLLF MVLDVTAKVG FGFILLRSRA
+ -+-++                 ++--++-- -

ILGDTEAPEP SAGAEASAAD
```

Fig. 1. Functional Sites of *bacteriorhodopsin*

References

1. Attwood, T. K. and Parry-Smith, D. J.: *Introduction to bioinformatics.* Longman (1999)
2. Bairoch, A, Bucher, P., and Hofmann, K.: The PROSITE database, its status in 1997, *Nucl. Acids Res.*, Vol.24, pp.217–221 (1997)
3. Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (eds.).: *Advances in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press (1996)
4. Futai, M. (ed.): *Biomembrane Engineering* (in Japanese), Maruzen (1991)
5. Ishikawa, T., Mitaku, S., Terano, T., Hirokawa, T., Suwa, M., and Seah, B-C.: Building A Knowledge-Base for Protein Function Prediction using Multistrategy Learning, In *Proceedings of Genome Informatics Workshop 1995*, pp.39–48 (1995)
6. Ishikawa, T., Terano, T. and Numao, M.: A Computation Method of Relative Least General Generalization Using Literal Association and MDL Criteria, Journal of Japanese Society for Artificial Intelligence (in Japanese), Vol.14, No. 2, pp.326–333 (1999)
7. Muggleton, S. and Feng, C.: Efficient Induction of Logic Programs, In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, Ohmsha (1990)
8. Muggleton, S., King, R., and Sternberg, M.: Protein Secondary Structure Prediction using Logic., *Protein Engineering*, Vol.5, pp.647–657 (1992)
9. Muggleton, S. and De Raedt, L.: Inductive Logic Programming: Theory and Methods, *The Journal of Logic Programming*, Vol.19, pp.629–679 (1994)
10. Muggleton, S.: Inverse Entailment and Progol. *New Generation Computing*, Vol.13, pp.245–286 (1995)
11. Quinlan, R.: Learning Logical Definition from Relations, *Machine Learning*, Vol.5, pp.239–266 (1990)

Mining Web Transaction Patterns in an Electronic Commerce Environment

Ching-Huang Yun and Ming-Syan Chen

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
chyun@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw

Abstract. In this paper, we explore a new data mining capability which involves mining Web transaction patterns for an electronic commerce (EC) environment. We propose an innovative mining model that takes both the traveling patterns and purchasing patterns of customers into consideration. First, we develop algorithm WR to extract meaningful Web transaction records from Web transactions so as to filter out the effect of irrelevant traversal sequences. Second, we devise algorithm WTM for determining the large transaction patterns from the Web transaction records obtained.

1 Introduction

Some existing electronic commerce environments [1][2], Web pages are usually designed as shop-windows. Customers can visit these Web pages and make Web transactions through the Web interface. It is known that mining information from such an EC system can provide very valuable information on consumer buying behavior and the quality of business strategies can then be improved [4]. Consequently, we shall explore in this paper a new data mining capability which involves mining *Web transaction patterns* for an EC environment.

First, for each Web transaction, we develop algorithm *Web-transaction-Record (WR)* algorithm, to extract meaningful Web transaction records from a given Web transaction. Each Web transaction record is represented by the form: *¡path: a set of purchases¿*, where a purchase, denoted by N(i), means that item i was purchased in node N along the path. After all the Web transaction records are derived from Web transactions, algorithm *Web Transaction Mining (WTM)* is developed for determining the large transaction patterns from the Web transaction records. Similarly to DHP [6], algorithm WTM utilizes the purchasing patterns for the candidate transaction pattern generation in the pattern discovering procedure. An illustrative example is given for the algorithm proposed.

This paper is organized as follows. Preliminaries are given in Section 2. Algorithms for determined Web transaction patterns is described in Section 3. This paper concludes with Section 4.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 216–219, 2000.
© Springer-Verlag Berlin Heidelberg 2000

2 Preliminaries

Let N = {n_1, n_2,..., n_g} be a set of nodes in the EC environment and I = {i_1, i_2,..., i_h} be a set of items sold in the system. We then have the following definitions.

Definition 1. *Let* ¡$s_1 s_2 ... s_y$: $n_1\{i_1\}$, $n_2\{i_2\}$,..., $n_x\{i_x\}$¿ *be a transaction pattern, where* $i_m \subseteq I$ *for* $1 \leq m \leq x$, *and* $\{n_1, n_2, ..., n_x\} \subseteq \{s_1, s_2, ..., s_y\} \subseteq N$. *Then,* ¡$s_1 s_2 ... s_y$: $n_1\{i_1\}$, $n_2\{i_2\}$,..., $n_x\{i_x\}$¿ *is said to* **pattern-contain** *a transaction pattern* ¡$w_1 w_2 ... w_q$: $r_1\{t_1\}$, $r_2\{t_2\}$,..., $r_p\{t_p\}$¿ *if and only if* $\{s_1 s_2 ... s_y\}$ *contains* $\{w_1 w_2 ... w_q\}$ *and* $\{n_1\{i_1\}, n_2\{i_2\}, ..., n_x\{i_x\}\}$ *contains* $\{r_1\{t_1\}, r_2\{t_2\}, ..., r_p\{t_p\}\}$.

Definition 2. *A Web transaction is said to* **pattern-contain** ¡$w_1 w_2 ... w_q$: $r_1\{t_1\}$, $r_2\{t_2\}$,..., $r_p\{t_p\}$¿ *if one of its Web transaction records pattern-contains* ¡$w_1 w_2 ... w_q$: $r_1\{t_1\}$, $r_2\{t_2\}$,..., $r_p\{t_p\}$¿.

A Web transaction consists of a set of purchases along the corresponding nodes in its traversal path. A transaction pattern is a large transaction pattern if there is a sufficient number of Web transactions pattern-containing it. It is worth mentioning that by taking both the traveling patterns and purchasing patterns into consideration, the problem of mining Web transaction patterns is in nature different from those addressed in prior works [3][5].

3 Algorithms for Web Transaction Patterns

In general, a Web transaction, generated from electronic commerce services, consists of a traversal path and a list of items purchased along the path. Given a Web transaction of a customer, algorithm WR is devised to derive Web transaction records to capture the customer traveling and purchasing behaviors in an EC environment.

Algorithm WR

Step 1. For each Web transaction, constructing a customer transaction tree by mainly incorporating customer transaction records, which correspond to nodes with purchases, as branches. Each customer transaction record includes the traversal path and the items purchased in the last node of this path.

Step 2. Determining all the Web transaction records by traversing customer transaction tree obtained in Step 1 in a depth-first manner.

Step 3. Storing the Web transaction, including the Web transaction records and the corresponding WT_ID, into the database.

Algorithm WTM

A transaction pattern with k-purchase ¡$s_1 s_2 ... s_p$: $n_1\{i_1\}$, $n_2\{i_2\}$,..., $n_k\{i_k\}$¿ is called a large k-transaction pattern, if there are a sufficient number of Web transactions pattern-containing it. Let C_k be a candidate set of large k-transaction patterns and T_k represent the set of the large k-transaction patterns. Similarly to DHP [6], WTM utilizes large transaction patterns for generating candidate transaction patterns. Furthermore, WTM employs a sophisticated hash tree, called

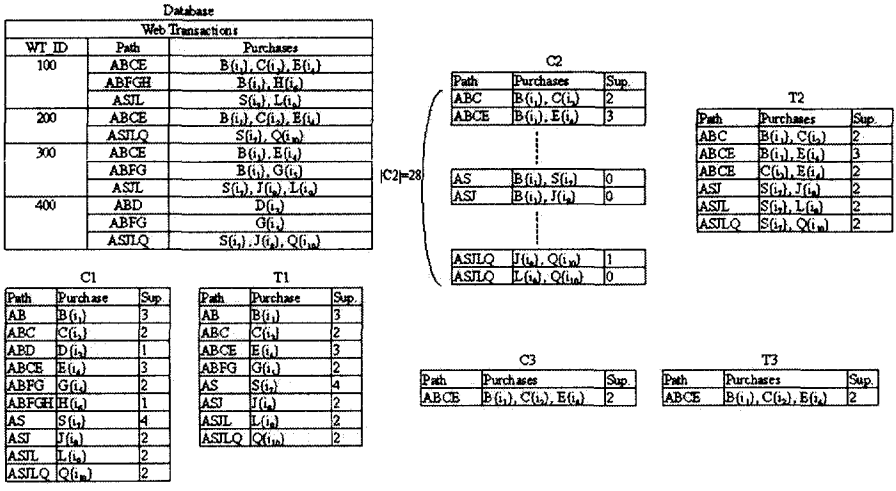


Fig. 1. *An illustrative example for mining Web transaction patterns*

Web transaction tree, to store candidate transaction patterns. WTM hashes not only each node but also each purchase in the path. According to each Web transaction record of a Web transaction, the support of a candidate transaction pattern is determined by the number of Web transactions that pattern-contain this candidate transaction pattern. WTM then obtains the large transaction patterns by destructing the Web transaction tree. Consider the example scenario in Figure 1. In the first pass, where WTM constructs the Web transaction tree by hashing each Web transaction record to construct the Web transaction tree and counts the support of individual purchases. Then, WTM destructs the Web transaction tree for deriving T_1, the set of large 1-transaction patterns. In each subsequent pass, WTM starts with the large transaction patterns found in the previous pass for generating new candidate transaction patterns to be stored in a Web transaction tree. Then, WTM proceeds to the counting of supports and finally reaches the generation of large transaction patterns.

After all large transaction patterns are obtained, one can derive the Web-transaction association rules from the large transaction patterns. In this example, ¡ABCE : B{i_1}, C{i_2}, E{i_4}¿ is one large 3-transaction pattern with support = 2 and ¡AB : B{i_1}¿ is one large 1-transaction pattern with support = 3. As a result, we can derive one Web-transaction association rule ¡ABCE : B{i_1} $\Longrightarrow$ C{i_2}, E{i_4}¿ with the support equal to support(¡ABCE : B{i_1}, C{i_2}, E{i_4}¿) = 2 and the confidence equal to $\frac{support(<ABCE:B\{i_1\},C\{i_2\},E\{i_4\}>)}{support(<AB:B\{i_1\}>)} = 67\%$.

4 Conclusion

In this paper, we explored a new data mining capability which involves mining Web transaction patterns. First, we developed algorithm WR to extract meaningful Web transaction records from Web transactions so as to filter out the

effect of irrelevant traversal sequences. Second, we devised algorithm WTM for determining the large transaction patterns from the Web transaction records obtained.

Acknowledgments

The authors are supported in part by the National Science Council, Project No. NSC 89-2219-E-002-007 and NSC 89-2213-E-002-032, Taiwan, Republic of China.

References

1. http://www.amazon.com/.
2. http://www.aol.com/.
3. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 478–499, September 1994.
4. A. G. Buchner and M. Mulvenna. Discovery Internet Marketing Intelligence through Online Analytical Web Usage Mining. *ACM SIGMOD Record*, 27(4):54–61, Dec. 1998.
5. M.-S. Chen, J.-S. Park, and P. S. Yu. Efficient Data Mining for Path Traversal Patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):209–221, April 1998.
6. J.-S. Park, M.-S. Chen, and P. S. Yu. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):813–825, October 1997.

Making Use of the Most Expressive Jumping Emerging Patterns for Classification

Jinyan Li[1], Guozhu Dong[2], and Kotagiri Ramamohanarao[1]

[1] Department of CSSE, The University of Melbourne, Parkville, Vic. 3052, Australia.
{jyli,rao}@cs.mu.oz.au
[2] Dept. of CSE, Wright State University, Dayton OH 45435, USA.
gdong@cs.wright.edu

Abstract. Classification aims to discover a model from training data that can be used to predict the class of test instances. In this paper, we propose the use of *jumping emerging patterns* (JEPs) as the basis for a new classifier called the *JEP-Classifier*. Each JEP can capture some crucial difference between a pair of datasets. Then, aggregating all JEPs of large supports can produce more potent classification power. Procedurally, the JEP-Classifier learns the *pair-wise features* (sets of JEPs) contained in the training data, and uses the *collective impacts* contributed by the *most expressive* pair-wise features to determine the class labels of the test data. Using only the most expressive JEPs in the JEP-Classifier strengthens its resistance to noise in the training data, and reduces its complexity (as there are usually a very large number of JEPs). We use two algorithms for constructing the JEP-Classifier which are both scalable and efficient. These algorithms make use of the *border representation* to efficiently store and manipulate JEPs. We also present experimental results which show that the JEP-Classifier achieves much higher testing accuracies than the association-based classifier of [8], which was reported to outperform C4.5 in general.

1 Introduction

Classification is an important problem in the fields of data mining and machine learning. In general, classification aims to classify instances in a set of test data, based on knowledge learned from a set of training data. In this paper, we propose a new classifier, called the *JEP-Classifier*, which exploits the discriminating power of *jumping emerging patterns* (JEPs) [4]. A JEP is a special type of EP [3] (also a special type of *discriminant rule* [6]), defined as an itemset whose support increases *abruptly* from zero in one dataset, to non-zero in another dataset — the ratio of support-increase being ∞. The JEP-Classifier uses JEPs exclusively, and is distinct from the CAEP classifier [5] which mainly uses EPs with *finite* support-increase ratios.

The exclusive use of JEPs in the JEP-Classifier is motivated by our belief that JEPs represent knowledge which discriminates between different classes more strongly than any other type of EPs. Consider, for example, the Mushroom dataset taken from the UCI data repository [1]. The itemset {ODOR = foul} is a JEP, whose support increases from 0% in the edible class to 55% in the poisonous class. If a test instance contains this particular EP, then we can claim with a very high degree of certainty that this

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 220–232, 2000.
© Springer-Verlag Berlin Heidelberg 2000

instance belongs to the poisonous class, and not to the edible class. In contrast, other kinds of EPs do not support such strong claims. Experimental results show that the JEP-Classifier indeed gives much higher prediction accuracy than previously published classifiers.

Example 1. This simplified example illustrates how JEPs are used in the JEP-Classifier. Consider two sets of training data, $\mathcal{D}_1$ and $\mathcal{D}_2$, such that all instances in $\mathcal{D}_1$ are of Class 1, and all instances in $\mathcal{D}_2$ are of Class 2. Let each instance be a subset of $\{a, b, c, d, e\}$ (see Table 1). **Question:** Which class should the test instance $\{a, b, c\}$ be classified as?

Table 1. Two simplified datasets containing 4 instances each.

$\mathcal{D}_1$				
a		c	d	e
a				
	b			e
	b	c	d	e

$\mathcal{D}_2$				
a	b			
		c		e
a	b	c	d	
			d	e

Answer: Class 2. **Rationale:** The test instance $\{a, b, c\}$ contains the JEP $\{a, b\}$ from $\mathcal{D}_1$ to $\mathcal{D}_2$, whose support in $\mathcal{D}_2$ is 50%. Furthermore, the remaining proper subsets of $\{a, b, c\}$ — namely, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, c\}$, and $\{b, c\}$ — appear in both classes of data with the same frequencies. These facts give us a higher confidence that the test instance should be classified as Class 2.

In general, a test instance T may contain several JEPs, and these EPs can favour different classes. Consider again the datasets in Table 1, this time with the test instance $T = \{a, b, e\}$. The instance T contains the following JEPs:

- the subsets $\{b, e\}$ and $\{a, e\}$, in favour of Class 1 with supports in $\mathcal{D}_1$ of, respectively, 50% and 25%;
- the subset $\{a, b\}$ in favour of Class 2, with a support in $\mathcal{D}_2$ of 50%.

We let all three JEPs contribute an *impact* equal to its support in its favoured class — the final decision is reached using the *collective impact*, obtained as the sum of the impacts of the individual JEPs, and choosing the class with the largest collective impact as the class of the test instance. It follows that the instance $\{a, b, e\}$ should be classified as Class 1, since the collective impact in favour of Class 1 ($50\% + 25\% = 75\%$) is larger than that of Class 2 (50%). This aggregation of the supports of JEPs is at the core of the JEP-Classifier.

There can be a large (e.g., 10^8) number of JEPs in the dense and high-dimensional datasets of a typical classification problem. Obviously, the naive approach to discovering all JEPs and calculating their collective impacts is too time consuming. For the JEP-Classifier, we utilize two *border*-based algorithms [3,4] to efficiently discover concise border representations of all JEPs from training dataset. The use of the border representation simplifies the identification of the *most expressive* JEPs. Intuitively, the most expressive JEPs are those JEPs with large support, which can be imagined as being at

the "frontier" of the set of JEPs. Itemsets which are proper subsets of the boundary itemsets are not JEPs, while itemsets which are proper supersets of the boundary itemsets must have supports *not larger* than the largest support of the boundary itemsets. These boundary JEPs represent the essence of the discriminating knowledge in the training dataset. The use of the most expressive JEPs strengthens the JEP-Classifier's resistance to noise in the training data, and can greatly reduce its overall complexity. Borders are formally defined in Section 3.

Example 1 above deals with a simple database containing only two classes of data. To handle the general cases where the database contains more classes, we introduce the concept of *pair-wise features*, which describes a collection of the discriminating knowledge of ordered pairs of classes of data. Using the same idea for dealing with two classes of data, the JEP-Classifier uses the collective impact contributed by the most expressive pair-wise features to predict the labels of more than two classes of data.

Our experimental results (detailed in Section 5) show that the JEP-Classifier can achieve much higher testing accuracy than previously published classifiers, such as the classifier proposed in [8], which generally outperforms C4.5, and the classifier in [5]. In summary, the JEP-Classifier has superior performance because:

1. Each individual JEP has sharp discriminating power, and
2. Identifying the most expressive JEPs and aggregating their discriminating power leads to very strong classifying ability.

Note that the JEP-Classifier can reach a 100% accuracy on any training data. However, unlike many classifiers, this does not lead to the usual overfitting problems, as JEPs can only occur when they are supported in the training dataset.

The remainder of this paper is organised as follows. In Section 2, we present an overall description of the JEP-Classifier (the learning phase and the classification procedure), and formally define its associated concepts. In Section 3, we present two algorithms for discovering the JEPs in a training dataset: one using a semi-naive approach, and the other using a border-based approach. These algorithms are complementary, each being useful for certain types of training data. In Section 4, we present a process for selecting the most expressive JEPs, which efficiently reduces the complexity of the JEP-Classifier. In Section 5, we show some experimental results using a number of databases from the UCI data repository [1]. In Section 6, we outline several previously published classifiers, and compare them to the JEP-Classifier. Finally, in Section 7, we offer some concluding remarks.

2 The JEP-Classifier

The framework discussed here assumes that the training database $\mathcal{D}$ is a normal relational table, consisting of N instances defined by m distinct attributes. An attribute may take categorical values (e.g., the attribute COLOUR) or numeric values (e.g., the attribute SALARY). There are q known classes, namely Class 1, $\cdots$, Class q; the N instances have been partitioned into q sets, $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_q$, according to their classes.

To encode $\mathcal{D}$ as a binary database, the categorical attribute values are mapped to *items* using bijections. For example, the two categorical attribute values, namely *red* and

yellow, of COLOR, are mapped to two items: (COLOR = *red*) and (COLOR = *yellow*). For a numeric attribute, its value range is first discretized into intervals, and then the intervals are mapped to items using an approach similar to that for categorical attributes. In this work, the values of numeric attributes in the training data are discretized into 10 intervals with the same length, using the so-called *equal-length-bin* method.

Let I denote the set of all items in the encoding. An *itemset* X is defined as a subset of I. The *support* of an itemset X over a dataset $\mathcal{D}'$ is the fraction of instances in $\mathcal{D}'$ that contain X, and is denoted $supp_{\mathcal{D}'}(X)$.

The most frequently used notion, JEPs, is defined as follows:

Definition 1. *The* JEPs *from $\mathcal{D}'$ to $\mathcal{D}''$, denoted* JEP$(\mathcal{D}', \mathcal{D}'')$, *(or called the JEPs of $\mathcal{D}''$ over $\mathcal{D}'$, or simply the JEPs of $\mathcal{D}''$ if $\mathcal{D}'$ is understood), are the itemsets whose supports in $\mathcal{D}'$ are zero but in $\mathcal{D}''$ are non-zero.*

They are named *jumping* emerging patterns (JEPs), because the supports of JEPs grow sharply from zero in one dataset to non-zero in another dataset.

To handle the general case where the training dataset contains more than two classes, we introduce the concept of *pair-wise features*.

Definition 2. *The* pair-wise features *in a dataset $\mathcal{D}$, whose instances are partitioned into q classes $\mathcal{D}_1, \cdots, \mathcal{D}_q$, consist of the following q groups of JEPs: those of $\mathcal{D}_1$ over $\cup_{j=2}^{q}\mathcal{D}_j$, those of $\mathcal{D}_2$ over $\cup_{j\neq 2}^{q}\mathcal{D}_j$, $\cdots$, and those of $\mathcal{D}_q$ over $\cup_{j=1}^{q-1}\mathcal{D}_j$.*

For example, if $q = 3$, then the pair-wise features in $\mathcal{D}$ consist of 3 groups of JEPs: those of $\mathcal{D}_1$ over $\mathcal{D}_2 \cup \mathcal{D}_3$, those of $\mathcal{D}_2$ over $\mathcal{D}_1 \cup \mathcal{D}_3$, and those of $\mathcal{D}_3$ over $\mathcal{D}_1 \cup \mathcal{D}_2$.

Example 2. The pair-wise features in $\mathcal{D}_1$ and $\mathcal{D}_2$ of Table 1 consist of the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$, $\{a, b\}$, $\{a, b, c\}$, $\{a, b, d\}$, $\{a, b, c, d\}$, and the JEPs from $\mathcal{D}_2$ to $\mathcal{D}_1$, $\{a, e\}$, $\{b, e\}$, $\{a, c, e\}$, $\{a, d, e\}$, $\{b, c, e\}$, $\{b, d, e\}$, $\{c, d, e\}$, $\{a, c, d, e\}$, $\{b, c, d, e\}$.

Note that we *do not enumerate* all these JEPs individually in our algorithms. Instead, we use *borders* to represent them. Also, the border representation mechanism facilitates the simple selection of the most expressive JEPs. The concept of border was proposed in [3] to succinctly represent a large collection of sets. (It will be reviewed later in section 3.)

Continuing with the above example, the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ can be represented by the border of $<\{\{a, b\}\}, \{\{a, b, c, d\}\}>$. Its *left bound* is $\{\{a, b\}\}$, and its *right bound* is $\{\{a, b, c, d\}\}$; it represents all those sets that are supersets of some itemset in its left bound, and are subsets of some itemset in its right bound. Obviously, $\{a, b\}$, the itemset in the left bound, has the *largest support* among all itemsets covered by the border. Similarly, the JEPs from $\mathcal{D}_2$ to $\mathcal{D}_1$ can be represented by two borders: $<\{\{a, e\}, \{c, d, e\}\}, \{\{a, c, d, e\}\}>$ and $<\{\{b, e\}, \{c, d, e\}\}, \{\{b, c, d, e\}\}>$. (Details will be given in Section 4.) Therefore, the *most expressive* JEPs in $\mathcal{D}_1$ and $\mathcal{D}_2$ are those in the set of $\{\{a, b\}, \{a, e\}, \{b, e\}, \{c, d, e\}\}$, the *union* of the left bounds of the three borders above. Observe that it is much smaller than the set of all JEPs.

In JEP-Classifier, the most expressive JEPs play a central role. To classify a test instance T, we evaluate the collective impact of only the most expressive JEPs that are subsets of T.

Definition 3. *Given a pair of datasets $\mathcal{D}'$ and $\mathcal{D}''$ and a test instance T, the collective impact in favour of the class of $\mathcal{D}'$ contributed by the most expressive JEPs of $\mathcal{D}'$ and of $\mathcal{D}''$ is defined as*

$$\sum_{X \in MEJEP(\mathcal{D}', \mathcal{D}'') \text{ and } X \subseteq T} supp_{\mathcal{D}'}(X),$$

where $MEJEP(\mathcal{D}', \mathcal{D}'')$ is the union of the most expressive JEPs of $\mathcal{D}'$ over $\mathcal{D}''$ and the most expressive JEPs of $\mathcal{D}''$ over $\mathcal{D}'$. The collective impact in favour of the class of $\mathcal{D}''$ is defined similarly.

The *classification procedure* of JEP-Classifier for a given test instance is a simple process as follows. Given a test instance T, the q collective impacts respectively in favour of the q classes are first computed. Then, the JEP-Classifier determines the class label as the class where T obtains the largest collective impact. When a tie occurs (i.e., the collective impacts obtained are equal), we can use popularities to break the tie.

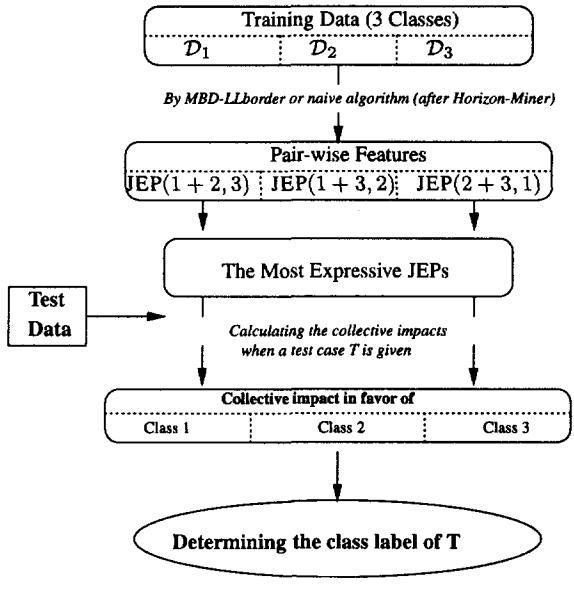


Fig. 1. JEP-Classifier working on a database with three classes of data.

Figure 1 depicts how the JEP-Classifier is built from the training data, and how it is then used to classify testing data, for the case when a database contains three classes of data. In this figure, JEP$(1 + 2, 3)$ represents the JEPs from $\mathcal{D}_1 \cup \mathcal{D}_2$ to $\mathcal{D}_3$, and similarly for JEP$(1 + 3, 2)$ and JEP$(2 + 3, 1)$. The HORIZON-MINER [4] and MBD-LLBORDER [3] algorithms, used to extract the pair-wise features from the training dataset, are outlined in Section 3. Determining the most expressive JEPs is discussed in Section 4.

3 Discovering the Pair-Wise Features

As the pair-wise features in $\mathcal{D}$ are defined as the JEPs over q pairs of datasets, we only need to consider how to discover the JEPs over one pair of datasets. Without loss of generality, suppose dataset $\mathcal{D}$ consists of only two classes of data $\mathcal{D}_1$ and $\mathcal{D}_2$, then the pair-wise features in $\mathcal{D}$ are the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ and the JEPs from $\mathcal{D}_2$ to $\mathcal{D}_1$. Now, we consider how to discover the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$.

The most naive way to find the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ is to check the frequencies, in $\mathcal{D}_1$ and $\mathcal{D}_2$, of all itemsets. This is clearly too expensive to be feasible. The problem of efficiently mining JEPs from dense and high-dimensional datasets is well-solved in [3][4]. The high efficiency of these algorithms is a consequence of their novel use of borders [3]. In the following subsections we present two approaches to discovering JEPs. The first approach is a semi-naive algorithm which makes limited use of borders, while the second approach uses an efficient border-based algorithm called MBD-LLBORDER [3].

3.1 Borders, Horizontal Borders, and HORIZON-MINER

A *border* is a structure used to succinctly represent certain large collections of sets.

Definition 4. *[3]. A border is an ordered pair $<\mathcal{L}, \mathcal{R}>$ such that each of $\mathcal{L}$ and $\mathcal{R}$ is an antichain collection of sets, each element of $\mathcal{L}$ is a subset of some element in $\mathcal{R}$, and each element of $\mathcal{R}$ is a superset of some element in $\mathcal{L}$; $\mathcal{L}$ is the* left bound *of the border, and $\mathcal{R}$ is its* right bound.

The collection of sets represented *by $<\mathcal{L}, \mathcal{R}>$ (also called the* set interval *of $<\mathcal{L}, \mathcal{R}>$) is*

$$[\mathcal{L}, \mathcal{R}] = \{Y \mid \exists X \in \mathcal{L},\ \exists Z \in \mathcal{R} \text{ such that } X \subseteq Y \subseteq Z\}.$$

We say that $[\mathcal{L}, \mathcal{R}]$ has *$<\mathcal{L}, \mathcal{R}>$ as its border, and that each $X \in [\mathcal{L}, \mathcal{R}]$ is covered by $<\mathcal{L}, \mathcal{R}>$.*

Example 3. The set interval of $<\{\{a, b\}\}, \{\{a, b, c, d, e\}, \{a, b, d, e, f\}\}>$ consists of twelve itemsets, namely all sets that are supersets of $\{a, b\}$ and that are subsets of either $\{a, b, c, d, e\}$ or $\{a, b, d, e, f\}$.

Definition 5. *The* horizontal border *of a dataset is the border $<\{\emptyset\}, \mathcal{R}>$ that represents all* non-zero support *itemsets in the dataset.*

Example 4. The horizontal border of $\mathcal{D}_1$ in Table 1 is $<\{\emptyset\}, \{\{a, c, d, e\}, \{b, c, d, e\}\}>$.

The simple HORIZON-MINER algorithm [4] was proposed to discover the horizontal border of a dataset. The basic idea of this algorithm is to select the maximum itemsets from all instances in $\mathcal{D}$ (an itemset is maximal in the collection C of itemsets if it has no proper superset in C). HORIZON-MINER is very efficient as it requires only one scan through the dataset.

3.2 The Semi-naive Approach to Discovering JEPs

The semi-naive algorithm for discovering the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ consists of the following two steps: (i) Use HORIZON-MINER to discover the horizontal border of $\mathcal{D}_2$; (ii) Scan $\mathcal{D}_1$ to check the supports of all itemsets covered by the horizontal border of $\mathcal{D}_2$; the JEPs are those itemsets with zero support in $\mathcal{D}_1$. The pruned SE-tree [3] can be used in this process to irredundantly and completely enumerate the itemsets represented by the horizontal border.

The semi-naive algorithm is fast on small databases. However, on large databases, a huge number of itemsets with non-zero support make the semi-naive algorithm too slow to be practical. With this in mind, in the next subsection, we present a method which is more efficient when dealing with large databases.

3.3 Border-Based Algorithm to Discover JEPs

In general, MBD-LLBORDER [3] finds those itemsets whose supports in $\mathcal{D}_2$ are $\geq$ some support threshold θ but whose support in $\mathcal{D}_1$ are less than some support threshold δ for a pair of dataset $\mathcal{D}_1$ and $\mathcal{D}_2$. Specially, this algorithm produces exactly all those itemsets whose supports are nonzero in $\mathcal{D}_2$ but whose supports are zero in $\mathcal{D}_1$, namely the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$. In this case, MBD-LLBORDER takes the horizontal border from $\mathcal{D}_1$ and the horizontal border from $\mathcal{D}_2$ as inputs. Importantly, this algorithm does not output all JEPs individually. Instead, MBD-LLBORDER outputs a family of borders in the form of $<\mathcal{L}_i, \mathcal{R}_i>$, $i = 1, \cdots, k$, to concisely represent all JEPs.

Unlike the semi-naive algorithm, which must scan the dataset $\mathcal{D}_1$ to discover the JEPs, MBD-LLBORDER works by manipulating the horizontal borders of the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. As a result, the MBD-LLBORDER algorithm scales well to large databases. This is confirmed by the experimental results in Section 5. The MBD-LLBORDER algorithm for discovering JEPs is described in detail in the Appendix.

4 Selecting the Most Expressive JEPs

We have given two algorithms to discover the pair-wise features from the training data $\mathcal{D}$: the semi-naive algorithm is useful when $\mathcal{D}$ is small, while the MBD-LLBORDER algorithm is useful when $\mathcal{D}$ is large. As seen in the past section, the MBD-LLBORDER algorithm outputs the JEPs represented by borders. These borders can represent very large collections of itemsets. However, only those itemsets with large support contribute significantly to the collective impact used to classify a test instance. By using only the most expressive JEPs in the JEP-Classifier, we can greatly reduce its complexity, and strengthen its resistance to noise in the training data.

Consider $\text{JEP}(\mathcal{D}_1, \mathcal{D}_2) \cup \text{JEP}(\mathcal{D}_2, \mathcal{D}_1)$, the pair-wise features in $\mathcal{D}$. Observe that $\text{JEP}(\mathcal{D}_1, \mathcal{D}_2)$ is represented by a family of borders of the form $<\mathcal{L}_i, \mathcal{R}_i>$, $i = 1, \cdots, k$, where the $\mathcal{R}_i$ are singleton sets (see the pseudo-code for MBD-LLBORDER in the Appendix). We believe that the itemsets in the left bounds, $\mathcal{L}_i$, are the most expressive JEPs in the dataset. The reasons behind this selection include:

- By definition, the itemsets in the left bound of a border have the largest supports of all the itemsets covered by that border because the supersets of an itemset X have smaller supports than that of X. Then, the most expressive JEPs cover more instances (at least equal) of the training dataset than the other JEPs.
- Any proper subset of the most expressive JEPs is not a JEP any more.

It follows that we can select the most expressive JEPs of $\text{JEP}(\mathcal{D}_1, \mathcal{D}_2)$ by taking the union of the left bounds of the borders produced by MBD-LLBORDER. This union is called the LEFT-UNION of $\text{JEP}(\mathcal{D}_1, \mathcal{D}_2)$. So, LEFT-UNION $= \cup \mathcal{L}_i$. Similarly, we can select the most expressive JEPs of $\text{JEP}(\mathcal{D}_2, \mathcal{D}_1)$. Combining the two LEFT-UNION, the most expressive pair-wise features in $\mathcal{D}$ are then constructed.

Algorithmically, finding LEFT-UNION can be done very efficiently. If the MBD-LLBORDER algorithm is used, then we simply use the left bounds of the borders it produces. In practice, this can be done by replacing the last line of the pseudo code of the MBD-LLBORDER algorithm in the Appendix with

return the union of the left bounds of all borders in EPBORDERS.

If the semi-naive algorithm is used, then LEFT-UNION can be updated as each new JEP is discovered.

Example 5. To illustrate several points discussed in this subsection, consider $\mathcal{D}_1$ and $\mathcal{D}_2$ from Table 1. The horizontal border of $\mathcal{D}_1$ is $<\{\emptyset\}, \{acde, bcde\}>$[1], and that of $\mathcal{D}_2$ is $<\{\emptyset\}, \{ce, de, abcd\}>$. The JEPs from $\mathcal{D}_2$ to $\mathcal{D}_1$ are represented by two borders $<\mathcal{L}_i, \mathcal{R}_i>$, $i = 1, 2$, namely $<\{ae, cde\}, \{acde\}>$ and $<\{be, cde\}, \{bcde\}>$. (The readers can use MBD-LLBORDER in the Appendix to derive these borders.)

The border $<\{ae, cde\}, \{acde\}>$ consists of the JEPs $\{ae, ace, ade, cde, acde\}$, while the border $<\{be, cde\}, \{bcde\}>$ consists of the JEPs $\{be, bce, bde, cde, bcde\}$. Note that the JEPs in the left bounds have the largest supports.

The LEFT-UNION of $\text{JEP}(\mathcal{D}_2, \mathcal{D}_1)$ is the union of the left bounds of the above two borders, namely $\{ae, cde\} \cup \{be, cde\} = \{ae, be, cde\}$.

5 Experimental Results

In this section we present the results of our experiments, where we run the JEP-Classifier on 30 databases (some contain up to 10 classes, some have up to 30162 instances, some have up to 60 attributes) taken from the UCI Repository of Machine Learning Databases [1]. These experiments were carried out on a 500MHz PentiumIII PC with 512M bytes of RAM. The accuracy was obtained using the methodology of ten-fold cross-validation [10] (but one fold was tested in census-income).

The experiment's pre-processes are: (i) download original datasets, say $\mathcal{D}$, from the UCI website; (ii) partition $\mathcal{D}$ into class datasets $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_q$; (iii) randomly shuffle $\mathcal{D}_i, i = 1, \cdots, q$; (iv) for each $\mathcal{D}_i$, choose the first 10% instances as the testing data and the remaining 90% as the training data. Repeatedly, choose the second 10% as the testing data, and so forth; (v) if there exist continuous attributes, discretize them by our

[1] For readability, we use $acde$ as shorthand for the set $\{a, c, d, e\}$.

equal-length-bin method in the training datasets first, and then map the intervals to the testing data. This step is used to convert the original training and testing data into the standard binary transactional data. (These executable codes are available from the authors on request.) After pre-processing, we followed the steps illustrated in Figure 1 to get the results. Alternatively, MLC++ technique [7] was also used to discretize continuous attributes in the glass, ionosphere, pima, sonar, and vehicle datasets. These testing accuracies are reported in Table 2. The main disadvantage of MLC++ technique is that it sometimes, for example in the liver dataset, produces many different instances with different labels into identical instances.

Table 2. Accuracy Comparison.

Datasets	#inst, attri, class	JEP-Cla.	CBA	C4.5rules	# JEPs	#CARs
anneal*	998, 38, 5	4.4	**1.9**	5.2	5059	65081
australian*	690, 14, 2	13.66	**13.2**	13.5	9806	46564
breast-w*	699, 10, 2	**3.73**	3.9	3.9	2190	399
census	**30162**, 16, 2	**12.7**	–	–	68053	–
cleve*	303, 13, 2	**15.81**	16.7	18.2	8633	1634
crx*	690, 15, 2	**14.06**	14.1	15.1	9880	4717
diabete*	768, 8, 2	**23.31**	24.7	25.8	4581	162
german*	1000, 20, 2	**24.8**	25.2	27.7	32510	69277
glass*	214, 9, 7	**17.4**	27.4	27.5	127	291
heart*	270, 13, 2	**17.41**	18.5	18.9	7596	624
hepatitis*	155, 19, 2	17.40	**15.1**	19.4	5645	2275
horse*	368, 28, 2	16.8	17.9	**16.3**	22425	7846
hypo*	3163, 25, 2	2.69	1.6	**0.8**	1903	493
ionosphere*	351, 34, 2	**6.9**	7.9	8.0	8170	10055
iris*	150, 4, 3	**2.67**	7.1	4.7	161	23
labor*	57, 16, 2	**8.67**	17.0	20.7	1400	313
liver	345, 6, 2	**27.23**	–	32.6	1269	–
lymph*	148, 18, 4	28.4	**18.9**	21.0	5652	2965
mushroom	**8124**, 22, 2	**0.0**	–	–	2985	–
nursery	**12960**, 8, 5	**1.04**	–	–	1331	–
pima*	768, 8, 2	**20.4**	26.9	24.5	54	2977
sick*	4744, 29, 2	2.33	2.7	**1.5**	2789	627
sonar*	208, 60, 2	**14.1**	21.7	27.8	13050	1693
soybean	47, 34, 4	**0.00**	–	8.3	1928	–
tic-tac-toe*	958, 9, 2	1.0	**0.0**	0.6	2926	1378
vehicle*	846, 18, 4	27.9	31.2	**27.4**	19461	5704
vote1*	433, 16, 2	8.53	6.4	**4.8**	5783	–
wine*	178, 13, 3	**6.11**	8.4	7.3	5531	1494
yeast*	1484, 8, 10	**33.72**	44.9	44.3	2055	–
zoo*	101, 16, 7	**4.3**	5.4	7.8	624	686

Table 2 summarizes the results. In this table, the first column lists the name of each database, followed by the numbers of instances, attributes, and classes in Column 2. The

third column presents the error rate of the JEP-Classifier, calculated as the percentage of test instances incorrectly predicted. Similarly, columns 4 and 5 give the error rate of, respectively, the CBA classifier in [8] and C4.5. (These results are the *best results* taken from Table 1 in [8]; a dash indicates that we were not able to find previous reported results). Column 6 gives the number of the most expressive JEPs used by the JEP-Classifier. The last column gives the number of CARs used in CBA.

These results raise several points of interest.

1. Our JEP-Classifier performed perfectly (100% or above 98.5% testing accuracy) on some databases (nursery, mushroom, tic-tac-toe, soybean).
2. Among the 25 databases marked with * (indicating results of both CBA and C4.5 are available) in table 2, the JEP-Classifier outperforms both C4.5 and CBA on 15 datasets; CBA wins on 5; and C4.5 wins on 5 (in terms of the testing accuracies).
3. For the databases (with bold font), they have much larger data sizes than the remaining databases. The JEP-Classifier performs well on those datasets.
4. For unbalanced datasets (having unbalanced numbers of instances for each class), the JEP-Classifier performs well. For example, nursery dataset contains 5 classes and have respectively 4320, 2, 328, 4266, and 4044 instances in each class. Interestingly, we observed that the testing accuracy by the JEP-Classifier was consistently around 100% for each class. For CBA, its support threshold was set as 1%. In this case, CBA would mis-classify all instances of class 2. The reason is that CBA cannot find the association rules in class 2.

Our experiments also indicate that the JEP-Classifier is fast and highly efficient.

- Building the classifiers took approximately 0.3 hours on average for the 30 cases considered here.
- For databases with a small number of items, such as the iris, labor, liver, soybean, and zoo databases, the JEP-Classifier completed both the learning and testing phases within a few seconds. For databases with a large number of items, such as the mushroom, sonar, german, nursery, and ionosphere databases, both phases required from one to two hours.
- In dense databases, the border representation reduced the total numbers of JEPs (by a factor of up to 10^8 or more) down to a relatively small number of border itemsets (approximately 10^3).

We also conducted experiments to investigate how the number of data instances affects the scalability of the JEP-Classifier. We selected 50%, 75%, and 90% of data instances from each original database to form three new databases. The JEP-Classifier was then applied to the three new databases. The resulting run-times shows a linear dependence on the number of data instances when the number of attributes is fixed.

6 Related Work

Extensive research on the problem of classification has produced a range of different types of classification algorithms, including nearest neighbor methods, decision tree induction, error back propagation, reinforcement learning, and rule learning. Most classifiers previously published, especially those based on classification trees (e.g., C4.5 [9],

CART [2]), arrive at a classification decision by making a sequence of micro decisions, where each micro decision is concerned with one attribute only. Our JEP-Classifier, together with the CAEP classifier [5] and the CBA classifier [8], adopts a new approach by testing groups of attributes in each micro decision. While CBA uses one group at a time, CAEP and the JEP-Classifier use the aggregation of many groups of attributes. Furthermore, CBA uses association rules as the basic knowledge of its classifier, CAEP uses emerging patterns (mostly with finite growth rates), and the JEP-Classifier uses jumping emerging patterns.

While CAEP has some common merits with the JEP-Classifier, it differs from the JEP-Classifier in several ways:

1. **Basic idea**. The JEP-Classifier utilizes the JEPs of large supports (the most discriminating and expressive knowledge) to maximize its collective classification power when making decisions. CAEP uses the collective classifying power of EPs with finite growth rates, and possibly some JEPs, in making decisions.
2. **Learning phase**. In the JEP-Classifier, the most expressive JEPs are discovered by simply taking the union of the left bounds of the borders derived by the MBD-LLBORDER algorithm (specialised for discovering JEPs). In the CAEP classifier, the candidate EPs must be enumerated individually after the MBD-LLBORDER algorithm in order to determine their supports and growth rates.
3. **Classification procedure**. The JEP-Classifier's decision is based on the collective impact contributed by the most expressive pair-wise features, while CAEP's decision is based on the normalized ratio-support scores.
4. **Predicting accuracy**. The JEP-Classifier outperforms the CAEP classifier in large and high dimension databases such as mushroom, ionosphere, and sonar. For small datasets such as heart, breast-w, hepatitis, and wine databases, the CAEP classifier reaches higher accuracies than the JEP-Classifier does. On 13 datasets where results are available for CAEP, the JEP-Classifier outperforms CAEP on 9 datasets.

While our comparison to CAEP is still preliminary, we believe that CAEP and JEP-Classifiers are complementary. More investigation is needed to fully understand the advantages offered by each technique.

7 Concluding Remarks

In this paper, we have presented an important application of JEPs to the problem of classification. Using the border representation and border-based algorithms, the most expressive pair-wise features were efficiently discovered in the learning phase. The collective impact contributed by these pair-wise features were then used to classify test instances. The experimental results have shown that the JEP-Classifier generally achieves a higher predictive accuracy than previously published classifiers, including the classifier in [8], and C4.5. This high accuracy results from the strong discriminating power of an individual JEP over a fraction of the data instances and the collective discriminating power by all the most expressive JEPs. Furthermore, our experimental results show that the JEP-Classifier scales well to large datasets.

As future work, we plan to pursue several directions. (i) In this paper, collective impact is measured by the sum of the supports of the most expressive JEPs. As alternatives, we are considering other aggregates, such as the squared sum, and adaptive methods, such as neural networks. (ii) In this paper, JEPs are represented by borders. In the worst case, the number of the JEPs in the left bound of a border can reach $C^N_{N/2}$, where N is the number of attributes in the dataset. We are considering the discovery and use of only some of the itemsets in the left bound, to avoid this worst-case complexity. (iii) In discovering JEPs using the MBD-LLBORDER algorithm, there are multiple uses of the BORDER-DIFF sub-routine, dealing with different borders. By parallelizing these multiple calls, we can make the learning phase of the JEP-Classifier even faster and more scalable.

Acknowledgements

We are grateful to Richard Webber for his help. We thank Bing Liu and Jiawei Han for useful discussions. We also thank the anonymous referees for helpful suggestions.

References

1. Blake, C. L., Murphy, P. M.: UCI Repository of machine learning database. [http://www.cs.uci.edu/ mlearn/mlrepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998)
2. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group (1984)
3. Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. Proceedings of ACM SIGKDD'99 International Conference on Knowledge Discovery & Data Mining, San Diego, USA, (1999) 43–52
4. Dong, G., Li, J., Zhang, X.: Discovering jumping emerging patterns and experiments on real datasets. Proceedings of the 9th International Database Conference (IDC'99), Hong Kong, (1999) 155–168
5. Dong, G., Zhang, X., Wong, L., Li, J.: CAEP: Classification by aggregating emerging patterns. DS-99: Second International Conference on Discovery Science, Tokyo, Japan, (1999)
6. Han, J., Fu, Y.: Exploration of the power of attribute-oriented induction in data mining. In: Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds): Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press (1996) 399–421
7. Kohavi, R. , John, G. , Long, R. , Manley, D. , Pfleger, K.: MLC++: a machine learning library in C++. Tools with artificial intelligence, (1994) 740–743
8. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. Proceedings of the 4th International Conference on Knowledge Discovery in Databases and Data Mining, KDD'98 New York, USA (1998)
9. Quinlan, J. R.: C4. 5: program for machine learning. Morgan Kaufmann (1992)
10. Salzberg, S. L.: On comparing classifiers: pitfalls to avoid and a recommended approach. Data Mining and Knowledge Discovery **1** (1997) 317 – 327

Appendix: MBD-LLBORDER for Discovering JEPs

Suppose the horizontal border of $\mathcal{D}_1$ is $<\{\emptyset\}, \{C_1, \cdots, C_m\}>$ and the horizontal border of $\mathcal{D}_2$ is $<\{\emptyset\}, \{D_1, \cdots, D_n\}>$. MBD-LLBORDER finds the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ as follows.

MBD-LLBORDER(horizontalBorder($\mathcal{D}_1$), horizontal-Border($\mathcal{D}_2$)) ;; *return all JEPs from* $\mathcal{D}_1$ *to* $\mathcal{D}_2$ *by multiple calls of* BORDER-DIFF
EPBORDERS$\leftarrow$ {};
for j from 1 to n **do**
  if some C_i is a superset of D_j then **continue**;
  $\{C'_1, \cdots, C'_m\} \leftarrow \{C_1 \cap D_j, \cdots, C_m \cap D_j\}$;
  RIGHTBOUND $\leftarrow$ all maximal itemsets in $\{C'_1, \cdots, C'_m\}$;
  add BORDER-DIFF($<\{\emptyset\}, D_j>, <\{\emptyset\}$, RIGHTBOUND$>$) into EPBORDERS;
return EPBORDERS;

BORDER-DIFF($<\{\emptyset\}, \{U\}>$, $<\{\emptyset\}, \{S_1, S_2, \cdots, S_k\}>$)
;; *return the border of* $[\{\emptyset\}, \{U\}] - [\{\emptyset\}, \{S_1, S_2, \cdots, S_k\}]$
initialize $\mathcal{L}$ to $\{\{x\} \mid x \in U - S_1\}$;
for $i = 2$ to k **do**
  $\mathcal{L} \leftarrow \{X \cup \{x\} \mid X \in \mathcal{L}, x \in U - S_i\}$;
  remove all itemsets Y in $\mathcal{L}$ that are not minimal;
return $<\mathcal{L}, \{U\}>$;

Note that given a collection $\mathcal{C}$ of sets, the *minimal* sets are those ones whose proper subsets are not in $\mathcal{C}$. For correctness and variations of BORDER-DIFF, the readers are referred to [3].

Mining Structured Association Patterns from Databases

Hirofumi Matsuzawa and Takeshi Fukuda

IBM Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato-shi,
Kanagawa 242-8502, Japan
{matuzawa,fukudat}@jp.ibm.com

Abstract. We consider the data-mining problem of discovering structured association patterns from large databases. A structured association pattern is a set of sets of items that can represent a two level structure in some specified set of target data. Although the structure is very simple, it cannot be extracted by conventional pattern discovery algorithms. We present an algorithm that discovers all frequent structured association patterns. We were motivated to consider the problem by a specific text mining application, but our method is applicable to a broad range of data mining applications. Experiments with synthetic and real data show that our algorithm efficiently discovers structured association patterns in a large volume of data.

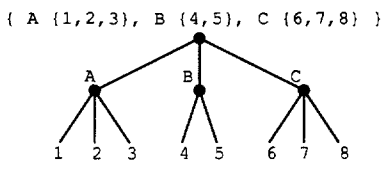
1 Introduction

Finding patterns in databases is the fundamental operation behind common data-mining tasks, including the mining of association rules [1,2,3,4,5,6,7] and sequential patterns [8,9]. For the most part, mining algorithms have been developed to discover very simple patterns such as sets or sequences of items. It is sometimes difficult for users to gain insight from such simple patterns.

While the patterns that mining algorithms generate are simple, the target data of interest often includes much more complex structures. In order to apply mining algorithms to complex data, what we usually do is to convert the data into a flat table, focusing on a fixed part of the original data structure and ignoring the rest. However, semi-structured data [10,11,12], such as data on the World Wide Web and collections of documents written in natural languages, contain structures that are not fixed beforehand, and such structures themselves may be interesting targets for knowledge discovery. If we can discover patterns with the same structure as a specified set of target data, such patterns will be intuitive and useful.

As a first step toward complex pattern mining, we adopt a set of sets of items as our form of patterns. Inner sets may have labels. The form of patterns delineates a tree structure with two levels (see Figure 1 for an example). We call such patterns *structured association* patterns. In this paper, we present an algorithm that discovers all frequent structured association patterns in databases.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 233–244, 2000.
© Springer-Verlag Berlin Heidelberg 2000

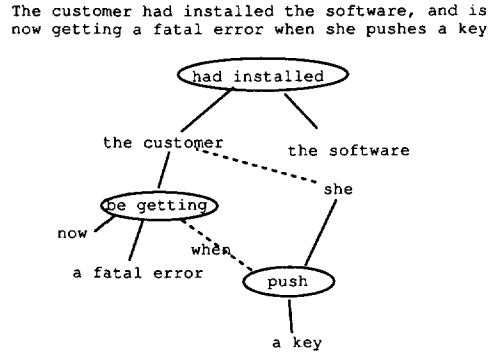


A, B, and C are labels of inner sets.

Fig. 1. Example of a pattern

predicate	arguments
install	{customer, software}
get	{customer, fatal error, now}
push	{she, key}

Fig. 2. Predicate-argument tuples



Circled words are predicative verbs.

Fig. 3. An example of a network

1.1 Motivating Example

Let us consider a call center of an enterprise. The call-takers input the summaries of customers' calls into the database as short documents in some natural language. We would like to extract from the database typical questions, requirements, complaints, and commendations. Since it is very hard for humans to discover such knowledge from a large volume of text data, it is natural to consider whether a pattern mining technique may help in the task.

One naive way is to find association rules or sequential patterns of words in the text by using standard pattern discovery algorithms. This will produce many patterns, but it is often difficult to recall the meaning of the original sentence from a discovered pattern, because the pattern does not contain any information on the roles of the words in the original sentence.

The progress of natural language processing technology has made it possible to automatically extract relationships between words by disambiguating the meanings of individual words. The technology is far from perfect, but it is very useful. Indeed, a typical machine translation method constructs a network structure from a sentence on the basis of the semantic relationship of words, performs some operations on the network, and then converts the network structure into a sentence in another language [13,14]. Figure 3 shows an example of a network structure obtained from a sentence. If we can find important patterns in the network structure, it will help us to realize our aim, since we can determine the meaning of each individual word in the network. Therefore, given a large collection of such networks, finding frequent subgraphs as patterns is a natural and interesting problem. But the computational complexity of this task seems to be very high. We therefore simplify the problem as follows.

We pay attention to "predicates" (or verbs), since they play the most significant semantic roles in a sentence. A predicate takes as "arguments" several words that have direct or indirect modification relationships with the predicate.

We treat such a group of words as a "predicate-argument tuple." For example, the sentence in Figure 3 has three groups, as shown in Figure 2. Although such groups do not carry all of the information in the original sentence, they are still much more informative than a simple set of individual words, and make it easier for users to recall the meaning of the original sentence.

A predicate-argument tuple can be represented by a set with a label, in which the predicate is a label and the arguments are elements of the set. Thus a sentence becomes a set of labeled sets. Therefore it is natural to consider the problem of mining frequent patterns that consist of labeled sets, such as { (install {software_A, driver_B}), (get {fatal, error}) }, from the text database.

Similar problems can be seen in the domain of molecular biology. Each individual molecule of nucleic acid is meaningless, but a sequence of them has some function, and a collection of such sequences may be the cause of an important biological phenomenon. Therefore, it would be useful to be able to find important collections in a DNA database.

2 Problem Statement

In this section, we give a formal statement of the problem, following those given in previous publications [1,2].

Definitions Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. A labeled set $g = (l, A)$ is a pair of a label $l \in \mathcal{I}$ and a set of items $A \subseteq \mathcal{I}$. We call labeled sets *groups*. Let $\mathcal{G} = \mathcal{I} \times \text{set}(\mathcal{I})$ denote the domain of groups. Let $\mathcal{D}$ be a set of transactions, where each transaction T is a set of groups ($T \subseteq \mathcal{G}$). Each transaction has a unique identifier, called a *TID*. Within a transaction, each group also has a unique identifier, called a *GID*. A *structured association pattern*, or pattern for short, is also a set of groups.

Pattern Matching We say that a group $g = (l_g, A_g)$ matches another group $h = (l_h, A_h)$, if $l_g = l_h$ and $A_g \subseteq A_h$. We say that a pattern $P = \{g_1, g_2, \ldots, g_k\}$ matches a transaction $T = \{h_1, h_2, \ldots, h_l\}$ if for each element $g_i \in P$ there exists a *distinct* group $h_j \in T$ such that g_i matches h_j. Note that these group-by-group matchings must be distinct. For example, a pattern $P_1 = \{(1, \{2\}), (1, \{4\})\}$ matches a transaction $T = \{(1, \{2, 3, 4\}), (1, \{4, 5\})\}$, but $P_2 = \{(1, \{2\}), (1, \{3\})\}$ does not, because both groups of P_2 match only the first group of T. We say that a pattern P' is a *subpattern* of P if P' matches P.

Support A pattern P has support s in the transaction set $\mathcal{D}$ if P matches s percent of transactions in $\mathcal{D}$. Even when a pattern matches a transaction more than once, the contribution of the transaction to the support count is only 1.

Problem Definition Given a set of transactions $\mathcal{D}$, the problem of mining structured association patterns is to generate all patterns that have support greater than the user-specified minimum support threshold (called *minsup*). We call a pattern *frequent* if the support of the pattern is greater than *minsup*. It is easy to construct an implication of the form $X \Rightarrow Y$ from a pattern P, where $X \cup Y = P$ and $X \cap Y = \emptyset$, in the same way as described in [2]. Therefore we do not discuss this point at present.

```
0)  MAIN(DataSet D) {
1)    // D is the input data set.
2)    // C_k is a set of candidate patterns of size k.
3)    // L_k is a set of frequent patterns of size k.
4)    C_1 ← {(l, ∅) | l ∈ I} ∪ {(*, {i}) | i ∈ I};
5)    k ← 1;
6)    while (C_k ≠ ∅) {
7)      COUNT(D, C_k);
8)      L_k ← {p ∈ C_k | p is frequent};
9)      C_{k+1} ← GENERATE-CANDIDATES(L_k);
10)     k ← k + 1;
11)   }
12)   return ⋃_k L_k;
13) }
```

Fig. 4. Outline of the Algorithm

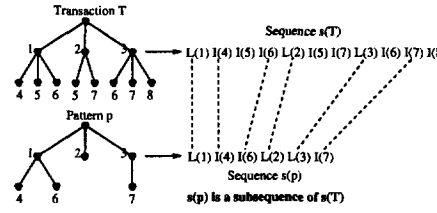


Fig. 5. Sequence matching (1)

We call the number of items in a group, including its label, the *size* of the group. In like manner, we call the sum of the group sizes in a pattern the *size* of the pattern. We refer to a pattern of size k as a k-pattern.

We assume that items in each group are kept sorted in lexicographic order, without loss of generality, because we are considering sets, not sequences. Similarly, we assume that groups in each transaction and pattern are kept sorted in lexicographic order of the labels of the groups. When $g_1 = (l_1, A_1)$ and $g_2 = (l_2, A_2)$ have the same label — that is, when $l_1 = l_2$ — the tie is resolved by looking at the lexicographic order of the first different elements of A_1 and A_2. Note that we allow multiple groups in a transaction to have the same labels; this is the tie case mentioned above. We can handle simple sets without labels by assigning a special label, *NULL*, to groups.

3 Algorithm

Figure 4 shows the outline of our algorithm, which is similar to that of other Apriori [2]-based pattern mining algorithms. First of all, we generate candidate patterns, each of which consists of a single item (line 4), where patterns of the form $(*, \{i\})$ have a special label "$*$" that matches any labels.

Each pass, say pass k, consists of two phases. First, the database is scanned to count the support of each candidate in C_k by calling a function named "COUNT" (line 7). The algorithm of COUNT uses a data structure similar to that of the subset function of Apriori [2], but it differs in that we need to handle the cases when some groups in a pattern have the same label. Next, the frequent patterns in L_k are used to generate the candidate patterns C_{k+1} for the $(k+1)$th pass by a function named "GENERATE-CANDIDATES" (line 9).

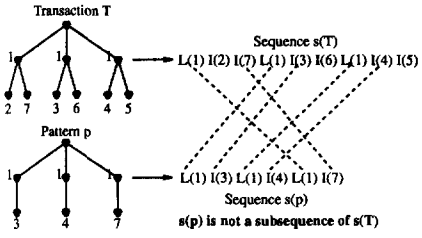
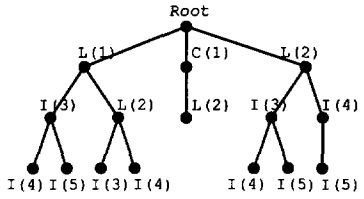


Fig. 6. Sequence matching (2)



A hash-tree storing $\langle L(1)I(3)I(4)\rangle$, $\langle L(1)I(3)I(5)\rangle$, $\langle L(1)L(2)I(3)\rangle$, $\langle L(1)L(2)I(4)\rangle$, $\langle C(1)L(2)\rangle$, $\langle L(2)I(3)I(4)\rangle$, $\langle L(2)I(3)I(5)\rangle$, and $\langle L(2)I(4)I(5)\rangle$.

Fig. 7. A hash-tree

3.1 Counting Support of Patterns

Given a set of groups p, we first convert it into a sequence of symbols $s(p)$. For example, when $p = \{(1,\{4,6\}), (2,\{\}), (3,\{7\})\}$, the corresponding sequence $s(p) = \langle L(1)\ I(4)\ I(6)\ L(2)\ L(3)\ I(7)\rangle$, where $L(*)$ denotes a symbol corresponding to a group label and $I(*)$ denotes a symbol corresponding to a group element. Let us assume that no transaction (and thus no pattern) contains more than one group with the same label. Then, as shown in Figure 5, we can decide whether a pattern p matches a transaction t by checking whether $s(p)$ is a subsequence of $s(t)$ and some constraint is satisfied[1]. Since Apriori uses a hash-tree to efficiently find all the patterns (itemsets) that are subsequences of a given transaction, it seems that we can apply a similar algorithm to our problem.

However, when some groups in a transaction may have the same group label, the above idea does not work. For example, Figure 6 shows a case in which a pattern p matches a transaction t even though $s(p)$ is not a subsequence of $s(t)$. Therefore in order to decide if a pattern p matches a transaction t, we need to determine if there exists a bipartite matching of $s(p)$ and $s(t)$ that covers all the elements of $s(p)$. Since we have many candidate patterns, and want to find from them all the patterns that match a given transaction, we need to solve multiple bipartite matching problems simultaneously. We use another hash-tree to solve those problems.

If a pattern p has multiple groups with same label, we treat them as a *cluster*, when p is converted into a sequence $s(p)$. For example, a pattern $\{(1,\{2,3\}), (2,\{1,3\}), (2,\{2,4\}), (3,\{2\})\}$ is translated into a sequence $\langle L(1),\ I(2),\ I(3),\ C(2),\ L(3),\ I(2)\rangle$, where the second and third groups of the pattern become a cluster. There are three types of elements: C (*cluster*), L (*label*), and I (*item*).

We store sequences of all patterns in C_k in the hash-tree. An interior node of the hash-tree is a hash table. When we add a sequence $s(p)$ to the hash-tree, we start from the root node and go down the tree by scanning the sequence. At

[1] If an element $I(x)$ in $s(t)$ is used in the matching, an element that corresponds to the group label of x must also be used.

a tree node at depth d, we decide which branch to follow by applying a hash function to the dth element of $s(p)$. Since this hash-tree is a kind of trie [15], any common prefixes of all patterns are shared in a path from the root node, and a path from the root to a leaf identifies the sequence of a pattern. Figure 7 shows an example.

Scanned transaction t is translated into a sequence $s(t)$. Starting from the root node, we find all the candidate patterns that match a transaction t. We apply the following procedure based on the type of node we are at:

- *Root node:* Apply a hash function to every item in $s(t)$, and recursively apply this procedure to the node in the corresponding bucket. For any pattern p that matches the transaction t, the first element of $s(p)$ must be in $s(t)$. By hashing on every item in $s(t)$, we ensure that we only ignore patterns that start with an element not in $s(t)$.
- *Interior node labeled I or L:* Assume that we reached this node by hashing on an element e in $s(t)$. Hash on each element that comes after e in $s(t)$ and recursively apply this procedure to the node in the corresponding bucket.
- *Interior node labeled C:* Construct bipartite graphs to find all candidate patterns that match the cluster of the transaction. Assume that we reached this node by hashing on an element e in $s(t)$. For each found pattern p that have a cluster that matches t, recursively apply this procedure to the tree of suffix pattern of p.
- *Leaf node:* We have now found a pattern that matches the transaction t.

Matching Clusters

Let us focus on a C-labeled node of the hash-tree. The node may have multiple — say, k — clusters each of which is a part of a pattern and shares a common prefix. Let $C = \{c_1, c_2, \ldots, c_k\}$ denote the set of the clusters. As we come to this node, the transaction t also contains a cluster $c(t)$ that has the same group label as c_is. Our goal is to find all the clusters in C that match $c(t)$.

In this paragraph, we consider the problem of deciding whether a single cluster c_i matches $c(t)$. Suppose that c_i has n groups and $c(t)$ has m groups ($n \leq m$). Let $c_i = \{g_1, g_2, \ldots, g_n\}$, and $c(t) = \{h_1, h_2, \ldots, h_m\}$. Consider a bipartite graph $G_i = (c_i, c(t); E_i)$ such that there exists an edge $(g_x, h_y) \in E_i$ if and only if $g_x \in c_i$ matches $h_y \in c(t)$. Figure 8 shows an example of a bipartite graph. c_i matches $c(t)$ if and only if there exists a bipartite matching in G_i that covers all g_i for $i = 1, \ldots, n$. Therefore, the problem of deciding whether c_i matches $c(t)$ can then be reduced to checking whether the size of the maximum bipartite matching in G_i is equal to n. This can be effectively solved by a classical algorithm[16]. In Figure 8, solid lines are the edges of the maximum bipartite matching. Since all groups in c_i are covered by the matching, c_i matches $c(t)$ in this case.

To effectively make all graphs $G_i = (c_i, c(t); E_i)$ for all $c_i \in C$, we construct another hash-tree beforehand, which stores all the candidate groups of c_i for $i = 1, \ldots, k$. For each group $h_j \in c(t)$, we identify all groups that match h_j by using the hash-tree, and add the corresponding edges to the graphs. After the

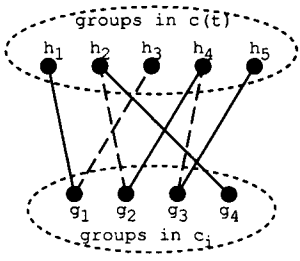


Fig. 8. Bipartite matching

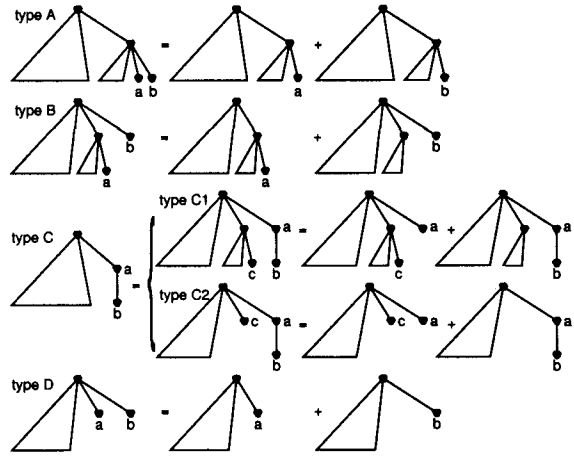


Fig. 9. Classification of patterns

above procedure, we know all the graphs, and hence we can find all clusters that match the transaction t.

3.2 Generating Candidates

The candidate generation function of Apriori joins $a = \{a_1, a_2, \ldots, a_k\}$ with $b = \{b_1, b_2, \ldots, b_k\}$, when they differ only in the last elements (i.e., $a_1 = b_1, \ldots, a_{k-1} = b_{k-1}$), to generate a new candidate $c = \{a_1, \ldots, a_{k-1}, a_k, b_k\}$. This method generates candidate patterns very efficiently even when there are a large number of patterns.

However, as for structured patterns, it may be impossible to drop the last elements because it may violate a structural constraint. For example, let us consider a pattern $p = \{(1, \{3\}), (1, \{4\})\}$. There are no pairs of proper patterns that differ only the last elements and that can be joined to generate p. In order to generate the pattern, we need to join $p_1 = \{(1, \{\}), (1, \{4\})\}$ and $p_2 = \{(1, \{3\}), (1, \{\})\} = \{(1, \{\}), (1, \{3\})\}$. Note that by joining p_1 with p_2, we can generate another pattern $p' = \{(1, \{\}), (1, \{3, 4\})\}$. Therefore candidate generation of structured patterns is not as simple as that of simple patterns.

Instead of thinking what k-sized patterns can be generated by joining two $(k-1)$-sized patterns, we inversely consider how we can split k-sized pattern into $(k-1)$-sized subpatterns. We classify patterns of size $k \geq 3$ into the following four types based on the last two elements, and split each of them into two subpatterns of size $(k-1)$ so that the k-sized pattern can be generated by joining the two subpatterns:

- **Type A:** The last two elements of the pattern are items of a single group.
- **Type B:** The last element is a group label, and the second last element is a item of the second last group.

- **Type C:** The last element is an item of the last group and the second last element is the group label. We further classify patterns of this type into Type C1 and C2 based on the third last elements.
- **Type D:**

Figure 9 illustrates this classification and how a pattern of each type can be constructed from its subpatterns. In this figure, a triangle represents a subpattern or subgroup. Because each element is either a label or an item, every pattern is classified into one of the above types. Therefore we can generate any candidate patterns by one of the expressions in the figure. Note that we must be careful to handle cases in which labels of the last and second last group are the same, because the order of the groups may change.

The GENERATE-CANDIDATES function takes as its argument the set of all frequent k-patterns L_k, and returns the set of candidate $(k+1)$-patterns C_{k+1}. The function works in two steps:

1. **Join Phase:** We generate candidate $(k+1)$-patterns by joining L_k with L_k. If a pattern $p_1 \in L_k$ is one of the patterns in the right-hand side of Figure 9, we find a pattern $p_2 \in L_k$ of the counterpart of p_1 and join p_1 and p_2 to generate a new candidate. Note that we do not have to search entire L_k for the counterparts of p_1, since we previously sort L_k in lexicographical order, and the range of the counterparts of p_1 is limited.
2. **Prune Phase:** We delete candidate patterns that have a k-subpattern whose support count is less than the minimum support. To effectively check whether all k-subpattern are in L_k, we use a hash-tree that stores L_k.

The above procedure is reminiscent of the candidate generation procedure for finding simple association rules; however details are quite different.

4 Experiments

To evaluate the effectiveness of the algorithm, we implemented it in C++, and performed several experiments on an IBM PC Server 330 running Windows NT 4.0 with a 333 MHz Intel Pentium II Processor and 320 MB of main memory. The data resided in the NT file system and was stored on a 9 GB SCSI 3.5" drive, with measured sequential throughput of about 8 MB/second.

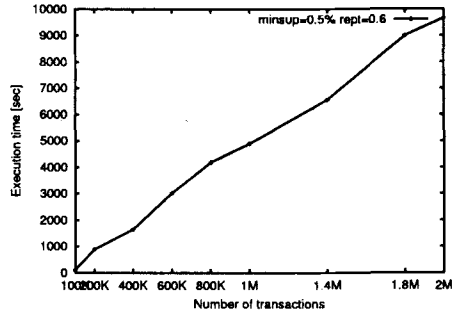
4.1 Synthetic Data

The synthetic data generator presented in [2] is commonly used for evaluating the performance of pattern-mining algorithms. We modified the generator so that it can generate structured transactions for our algorithm. Figure 10 shows the major parameter settings.

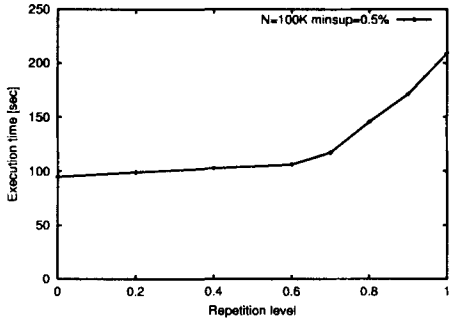
To see the scalability of the algorithm against the input size, we generated several datasets by changing the number of transactions from 100,000 (14.4 MB) to 2,000,000 (288 MB). We set the minimum support to 0.5 percent and the

Number of distinct labels	1,000
Number of distinct items	1,000
Average number of groups in a transaction	5
Average number of items in a group	5

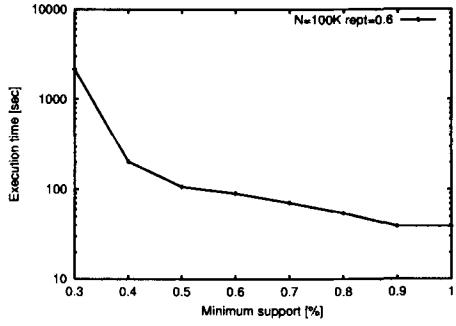
Fig. 10. Parameter Settings of Synthetic Datasets



(a) Data volume vs. execution time



(b) Repetition level vs. execution time



(c) Minsup level vs. execution time

Fig. 11. The experimental results on the synthetic datasets

repetition level to 0.6. Figure 11 (a) shows the relationship between the volume of input and the execution time of the algorithm. We can see that the execution time increases linearly in proportion to the number of transactions. When the number of transactions was 2,000,000, the algorithm discovered 6,236 patterns.

To assess how duplicated group labels affect the performance, we generated several datasets by changing a parameter called *repetition level*, which controls the average probability that a group has the same label as other groups in a transaction. We set the repetition level to 0, 0.2, 0.4, 0.6, 0.8, and 1.0. The average number of distinct group labels in a transaction decreases almost linearly in the repetition level, — that is, 5.0, 4.3, 3.6, 2.9, 2.2, and 1.6, respectively — while the average number of groups in a transaction is always 5. We set the number of transactions to 100,000 and the minimum support count to 500 (0.5%). Figure 11 (b) shows the result. While the execution time is almost linear in the repetition levels when the repetition level is relatively low, it starts to increase rapidly when the repetition level is more than 0.7.

We also conducted an experiment on how the minimum support threshold affects the performance. We fixed the number of transactions at 100,000 and the repetition level at 0.6. Figure 11 (c) shows the result. Note that the vertical scale is logarithmic. We can see a trend similar to that of other Apriori-based

Number of sentences	109,451
Number of predicative words	12,754
Number of non-predicative words	54,453
Average number of words in a sentence	4.23
Average number of groups in a sentence	1.48
Size	3.6MB

Fig. 12. Call center data

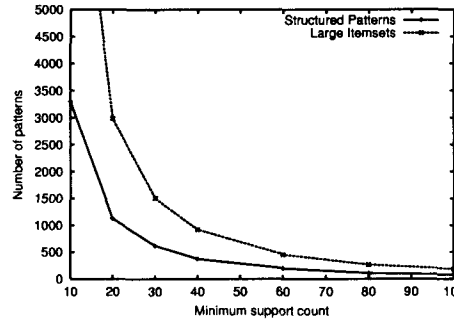


Fig. 13. Experimental results on the call center data

algorithms. When the minimum support is 0.3 (resp. 0.5, 0.7, 0.9, 1.0) percent, the number of frequent patterns is 93,074 (1264, 750, 602, 565).

4.2 Real Data

We applied our algorithm and a standard association rule mining algorithm to real data consisting of Japanese text obtained from a certain company's call center. For our algorithm, we preprocessed the original text data, using a shallow (and hence fast) natural language parser [17] developed at the authors' institution to convert all the sentences into sets of predicate-argument tuples as we explained in Subsection 1.1. We also converted all the sentences into flat sets of words so that an ordinary association discovery algorithm (which we call a conventional algorithm) can discover frequent sets of words as patterns. Figure 12 shows the characteristics of the data.

It takes less than two minutes for our algorithm to discover all the patterns, even when the minimum support count is only 5 (i.e., 0.005%). We do not compare the execution time, since the conventional algorithm used here runs on a different platform. As shown in Figure 13, the conventional algorithm discovers about twice as many patterns as our algorithm, even though there are combinatorially more structured patterns than simple itemsets. This experiment shows that more than half of the patterns discovered by the conventional method are bogus, because the patterns discovered only by the conventional algorithm consist of words that are used in very different contexts in the supporting sentences. Most patterns in which the analysts of the text data are interested have a support level of around 10 to 20. When the minimum support is at such a low level, the conventional algorithm generates too many patterns, half of which are bogus. Therefore, our method can significantly reduce the burden of text analysts who have been using ordinary pattern mining algorithms.

To see the effectiveness of our method from the view point of our motivating application, we examined *precision* and *recall*, commonly used for measuring effectiveness of information retrieval systems.

Our method			Conventional method	
Pattern	Recall (%)	Precision (%)	Recall (%)	Precision (%)
{(request {repair}) (say {})}	100	91.2	100	79.5
{(expire {guarantee}) (say {})}	100	100	100	87.2
{(ask {dealer}) (say {})}	97.9	100	100	76.2
{(attach {stereo-set, store})}	97.4	100	100	80.9
{(inspect {dealer}) (say {})}	96.8	93.8	100	57.4

Fig. 14. Recall and precision

We define recall and precision as follows. Suppose that we are given a set of semantic relationships of words, for instance, "someone purchases a car." We describe the relationships in pattern languages as { (purchase {car}) } for our method and { purchase, car } for the conventional method. We call a sentence *covered* by a pattern if the pattern matches the sentence. We call a sentence *relevant* if humans decide that the sentence contains the semantic relationships. Let X denote the number of covered sentences, let Y denote the number of relevant sentences, and let Z denote the number of covered and relevant sentences. Then recall is defined as Z/Y and precision is defined as Z/X.

We found recall and precision for several patterns. Figure 14 shows the result (the original words are in Japanese). Because patterns used by the conventional method (flat sets of words) certainly cover all relevant sentences, the recall of the conventional method is always 100 percent, but the precision is not very good. On the contrary, both recall and precision for our method are very high. Although the natural language parser sometimes makes mistakes, this result confirms the advantages of our approach.

5 Conclusions

We introduced the problem of mining structured association patterns from databases. A structured association pattern is a set of sets of items that can express a two-level structure in the target data. We presented an algorithm that efficiently discovers all frequent structured association patterns in a large database. We implemented the algorithm and evaluated it by using synthetic data and real text data. The method described in this paper can be used to solve problems in which each data element is not especially significant, but some combinations of elements are meaningful, and combinations of combinations of individual elements are interesting targets for discovery.

We are now developing a text mining system for call centers and evaluating our approach by using real text data. It is essential for text mining to utilize prior knowledge such as compound words and thesaural information. We plan to look into the problem of efficiently discovering structured patterns using prior knowledge. It would be interesting to investigate how much the accuracy of natural language processing affects the effectiveness of our approach. We also

plan to look into the problem of discovering more complex patterns from text and semi-structured data.

References

1. Rakesh Agrawal, Tomasz Imielinski, and Arum Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, May 1993.
2. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the International Conference on Very Large Data Bases*, pages 487–499, 1994.
3. Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *Proceedings of the International Conference on Very Large Data Bases*, pages 407–419, 1995.
4. Jiawei Han and Yongjian Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the International Conference on Very Large Data Bases*, pages 420–431, 1995.
5. Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 175–186, May 1995.
6. Sergay Brin, R. Motowani, Jeffery Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 255–264, 1997.
7. Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, May 1998.
8. Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the International Conference on Data Engineering*, 1995.
9. Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the International Conference on Extending Database Technology*, 1996.
10. Serge Abiteboul. Querying semi-structured data. In *Proceedings of the International Conference on Database Theory*, pages 1–18, January 1997.
11. Peter Buneman. Semistructured data. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, May 1997.
12. Ke Wang and Huiqing Liu. Schema discovery for semistructured data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 271–274, August 1997.
13. Sergei Nirenburg, editor. *Machine Translation*. Cambridge University Press, Cambridge, 1987.
14. Makoto Nagao, Jun-ichi Tsujii, and Jun-ichi Nakamura. The Japanese government project for machine translation. In Jonathan Slocum, editor, *Machine Translation Systems*, pages 141–186. Cambridge University Press, Cambridge, 1988.
15. Edward M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2):262–272, April 1976.
16. John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal of Computing*, 2(4):225–231, December 1973.
17. Tetsuya Nasukawa, Masayuki Morohashi, and Tohru Nagano. Customer claim mining: Discovering knowledge in vast amounts of textual data. Technical Report RT0319, IBM Tokyo Research Laboratory, May 1999.

Association Rules

Tao Zhang

Triada Ltd., 323B Vintage Park Dr., Foster City, CA 94404

Abstract. New association rules are presented for measure of association relationships between patterns. The new association rules are shown to not only measure three well-known association relationships correctly, but also satisfy other criteria for correct measure of association. Comparison with other measures is discussed both theoretically and experimentally. Applications in supervised mining of association rules and in pattern-driven multidimensional pattern analysis are presented.

1 Association Rules

Association rules have received much attention in the past (Agrawal et al. 1993a, Agrawal et al. 1993b, Agrawal and Srikant 1994, Klemettine et al. 1994, Mannila et al. 1994, Han and Fu 1995, Houtsman and Swami 1995, Park et al. 1995, Srikant and Agrawal 1995, Savasere et al. 1995, Agrawal et al. 1996a, Agrawal et al. 1996b, Cheung et al. 1996, Toivonen 1996, Lee et al. 1998, Meo et al. 1998, Wijsen and Meersman 1998, Silverstein et al. 1998). There are 2 fundamental problems in the study of association rules: association rules and mining association rules. The former is about measure of association relationships between patterns. The latter is about development of efficient techniques for finding interesting association rules, which may include development of additional measures (support for example) of pattern properties other than association, to identify interesting rules. Precise measure of association is sufficient for understanding association rules and is not sufficient for mining association rules. As the title suggested, measure of association is the main subject in this paper. An association rule is a measure of the amount of association relationship between patterns (or information events) in quantity, similar to the information entropy which is a measure of the amount of information in quantity. The simplest association pattern is a mathematical expression of transactional relationship between two patterns A and B, instead of sequential relationship. Unlike transactional correlation which has no direction, the transactional association has direction, meaning A associated with B is not equal to B associated with A. Therefore, a correct measure of association should be not only a transactional measure, but also sensitive to direction. Currently, two methods are well recognized for measure of association: conditional measure (or conditional rule) and the χ^2 measure (or contingency table). A conditional measure is a sequential measure of typical if-then sequence. While the χ^2 test is a transactional measure, it is not sensitive to direction because it is a symmetric measure. A fundamental

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 245–256, 2000.
© Springer-Verlag Berlin Heidelberg 2000

property of a correct measure of association is that it has to be transactional measure sensitive to direction. We developed a new formula for association rules (T. Zhang, US patent pending). Our starting point is to recognize the difference between association and disassociation, similar to the difference between attractive and repulsive forces. If probability of co-occurrence $P(A|B)$ for patterns A and B is larger than probability of no co-occurrence $P(A|\bar{B})$, the relationship of A associated with B is association (attractive). Otherwise, the relationship is disassociation (repulsive). The association relationship is described by $P_A(A \Rightarrow B) = 1 - P(A|\bar{B})/P(A|B)$, if $P(A|B) > P(A|\bar{B})$. The disassociation relationship is described by $P_D(A \Rightarrow B) = P(A|B)/P(A|\bar{B}) - 1$, if $P(A|B) \leq P(A|\bar{B})$. Combining the two formulas, we obtain

$$P(A \Rightarrow B) = \frac{P(A|B) - P(A|\bar{B})}{MAX[P(A|B), P(A|\bar{B})]} = \frac{P(AB) - P(A)P(B)}{MAX[P(AB)(1 - P(B)), P(B)(P(A) - P(AB))]} \quad (1)$$

where association pattern $A \Rightarrow B$ describes transactional association of pattern A with pattern B. $P(A)$, $P(B)$, and $P(AB)$ are probabilities for patterns A, B, and $A \wedge B$ respectively. It is noted that the probabilities are approximated by corresponding frequencies in a database. The approximation is exact when the number of records in the database is infinite large. Our association rule measures direction since the above formula is not symmetric about A and B. In order to compare our association rule with other measures of association, we present formulas for these measures as well. Association measured by a conditional rule is given by

$$P(A \Rightarrow B) \approx P(B|A) = \frac{P(AB)}{P(A)}. \quad (2)$$

Association measured by the χ^2 test is given by

$$\frac{\chi^2}{N} = \frac{[P(AB) - P(A)P(B)]^2}{P(A)P(B)} + \frac{[P(A\bar{B}) - P(A)P(\bar{B})]^2}{P(A)P(\bar{B})} + \frac{[P(\bar{A}B) - P(\bar{A})P(B)]^2}{P(\bar{A})P(B)} + \frac{[P(\bar{A}\bar{B}) - P(\bar{A})P(\bar{B})]^2}{P(\bar{A})P(\bar{B})} \quad (3)$$

where N is the number of records. Here, we have normalized the result in a range between 0 and 1. 0 means complete independence while 1 represents complete dependence. The above formula may reduce to the following

$$\frac{\chi^2}{N} = \frac{[P(AB) - P(A)P(B)]^2}{P(A)P(B)(1 - P(A))(1 - P(B))}. \quad (4)$$

The above formula is symmetric about A and B. Therefore, it does not measure association direction. Furthermore, measure of interest is used to identify which one of 4 pairs gives the most significant contribution since the above formula aggregates 4 related pairs (Silverstein et al. 1998). Because of being insensitive to association direction, the χ^2 test is ruled out for correct measure of association relationship.

2 Comparison with Existing Measures of Association

In the above description of association rules, we argued that the current association rules approximated by either conditional rules or the χ^2 test do not measure associations accurately. Now, we prove it. A fundamental test for a correct measure of associations is a test of three well-known associations: complete association $P(AB) = P(A)$, complete disassociation $P(AB) = 0$, and complete random association or complete independence $P(AB) = P(A)P(B)$. A correct measure should give a definitive result for each of the three association tests without uncertainty. In another word, the result for each test should be a constant independent of $P(A)$ and/or $P(B)$, rather than a function of $P(A)$ and/or $P(B)$. For the complete association, pattern A is always associated with pattern B. The normalized association should be a maximum or 100% regardless what $P(A)$ and $P(B)$ are. Our new association rule gives the following result $P(A \Rightarrow B) = \frac{P(A)(1-P(B))}{MAX[P(A)(1-P(B)),0]} = 1$. Conditional rule gives $P(B|A) = \frac{P(AB)}{P(A)} = 1$. Both association rule and conditional rule pass the test. On the other hand, the χ^2 test gives $\frac{\chi^2}{N} = \frac{P(A)[1-P(B)]}{P(B)[1-P(A)]}$ which fails the test because the result is a function of $P(A)$ and $P(B)$, instead of a constant. For the complete disassociation test, pattern A is never associated with pattern B. The normalized disassociation should be a maximum or 100% no matter what $P(A)$ and $P(B)$ are. Our new association rule gives $P(A \Rightarrow B) = -\frac{P(A)P(B)}{MAX[0,P(A)P(B)]} = -1$. Conditional rule gives $P(B|A) = \frac{P(AB)}{P(A)} = 0$. Our new association rule and conditional rule pass the second test. The χ^2 measure fails to pass the test because it gives $\frac{\chi^2}{N} = \frac{P(A)P(B)}{[1-P(A)][1-P(B)]}$ which is a function of $P(A)$ and $P(B)$, instead of a constant. For the complete random association, pattern A is randomly associated with pattern B. Our new association rule gives $P(A \Rightarrow B) = \frac{0}{P(A)P(B)[1-P(B)]} = 0$. The χ^2 test gives $\frac{\chi^2}{N} = \frac{0}{P(A)P(B)[1-P(A)][1-P(B)]} = 0$. This time, association rule and dependence rule (χ^2 test) pass the test. Conditional rule gives $P(B|A) = P(B)$ which is a function of $P(B)$, instead of a constant. Therefore, conditional rule fails the independence test. Our new association rule is the only one that passes all three tests.

3 Other Possible Measures of Association

Having shown that our association measure passes successfully a test of three well-known association relationships and that both conditional rules and the χ^2 test fail to pass the same test, we may ask whether our new measure of association is the only correct measure of association relationships between patterns. This question is difficult to answer directly without rigorous proof. Instead, we look at other measures of association, that may pass the same test. One such measure may be obtained by modifying conditional measure. The main problem for a conditional rule to measure association relationship is that it can not distinguish

association from disassociation except two extreme situations when conditional confidence is either 1 or 0. This is due to absence of comparison of a given conditional confidence with unconditional probability of a target. If the former is larger than the latter, the relationship is association. If the former is less than the latter, the relationship is disassociation. Otherwise, it is random association. Therefore, we modify conditional measure by subtracting the latter from the former, or

$$P_m = P(B|A) - P(B). \tag{5}$$

Positive results describe association relationships while negative results represent disassociation relationships. If results are positive, normalize the modified measure by dividing the above formula by 1-P(B). The normalized formula describes association. Otherwise, normalize the modified measure by dividing the formula by P(B). The resulting formula describes disassociation relationship. The normalized association formula becomes

$$P_a = \frac{P(B|A) - P(B)}{1 - P(B)} = \frac{P(AB) - P(A)P(B)}{P(A)(1 - P(B))}. \tag{6}$$

The normalized disassociation formula becomes

$$P_d = \frac{P(B|A) - P(B)}{P(B)} = \frac{P(AB) - P(A)P(B)}{P(A)P(B)}. \tag{7}$$

The above two formulas measure the three well-known associations correctly indeed. However, there are other criteria or properties for test of correct association measure. One is measure of association direction as indicated before. It is seen that the second formula above is symmetric about A and B. Therefore, it does not measure direction necessary for association relationship. Another property is symmetry between association and disassociation, similar to the symmetry between attractive and repulsive forces. Such symmetry does not exist in the above two equations while our association and disassociation do have such symmetry property. But, the most important criteria is a clear, unique, and independent interpretation of the measured association relationship. Results from the above two formulas do not have such interpretation. In contrast, results from our association measure have adequate interpretation. The interpretation is that it measures the strength of probability of co-occurrence relative to probability of no co-occurrence for association, and the strength of probability of no co-occurrence relative to probability of co-occurrence for disassociation. This interpretation becomes more evident if we rewrite our association formula as $P(A|B)/P(A|\bar{B}) = 1/(1 - P(A \Rightarrow B))$ for association and $P(A|\bar{B})/P(A|B) = 1/(1 + P(A \Rightarrow B))$ for disassociation. For $P(A \Rightarrow B) = 0.5, 0.75, 0.8, 0.9, 0.95$, or 0.98, pattern A is 1,3,4,9,19,49 times more likely to appear with pattern B than not, respectively. Similarly, pattern A is 1,3,4,9,19,49 times more likely to appear in the absence of pattern B than they appear together for $P(A \Rightarrow B) = -0.5, -0.75, -0.8, -0.9, -0.95$, or -0.98 respectively. Each value has unique and independent interpretation. A symmetric correspondence between association and disassociation values exists

for all values using our measure. Similar correspondence does not exist between association and disassociation values using the two formulas modified from conditional measure. To see the point more clearly, we express the modified formulas as multiplication of our new association and conditional confidence, or $P_a = P(A \Rightarrow B)P(B|A)$ and $P_d = P(A \Rightarrow B)P(\bar{B}|A)$, which indicate that association relationships that have the same value in our measure have different values if the corresponding conditional confidence values differ from each other. This means that one of the two measures must be incorrect because of multiple measured results for the same association. Since the second formula modified from conditional measure does not measure disassociation direction and both formulas modified from conditional measure fail to have proper interpretation of association, formulas modified from conditional measure do not measure association accurately.

4 Supervised Mining of Association Rules

Having presented new association rules and theoretical comparison with other measures of association relationship, we discuss briefly mining association rules before we present experimental comparison and applications. Like mining other patterns and relationships, mining association rules is to find interesting association rules, which may require additional measures to define interestingness independing of association relationship. Such measures may change from one to another, depending on objectives for mining interesting rules. A well-established measure is support which measures how often patterns co-occur. We want to emphasize that support does not measure any relationship between patterns, but rather measures frequency of occurrence or co-occurrence of patterns. Frequency is about how often patterns co-occur while relationship is about how co-occurring patterns relate to patterns not co-occurring. Therefore, support or any other measure that does not measure association relationship is not part of association rules, and only makes sense in mining association rules. There have been significant work done in mining association rules in an unsupervised way such as regular market basket analysis. Here, we present an approach for mining association rules in a supervised manner. Supervised mining of association rules is to define a target (business problem or business opportunity), and to identify interesting patterns associated with the target. The supervised mining can be very useful in business pattern analysis in databases. For example, Coca-Cola would be much more interested in the relationships between Coke and other products, and between products of their competitors and other products, than relationships between non-soft-drink products in POS data. Here, a target may be Coke products or their competitors' products. In general, business people at higher level want to know how patterns existing in their business database relate to a well-defined business problem or opportunity. It is much easier to perform supervised mining of association rules than unsupervised mining because only rules describing association relationships between a target pattern and other patterns need to be obtained. Similar to unsupervised mining of asso-

ciation rules, support measure may be used for identification of interesting rules. However, we re-normalize conventional support by multiplying the total number of records divided by the total counts of a given target pattern in the study of association relationships between the target pattern and other patterns. In this way, the modified support values represent contributions, in percentage, to the target (problem or opportunity) from other patterns. We use this re-normalized support measure in the following applications.

5 Experimental Results

In this section, we apply our new association rules to real-world examples in comparison with conditional rules. No experimental comparison with the χ^2 test is given because of being unable to measure association direction by the χ^2 test. Comparison of association rules with the χ^2 test would be equivalent to comparison of a vector with a scalar, which is meaningless. We start with a simple POS transactional example shown in Table 1 (Mannila 1998). Table 1 shows 10 rows of transactions of a POS data set. Find out association rules for association patterns $mustard \wedge sausage \wedge beer \Rightarrow chips$ and $mustard \wedge sausage \wedge beer \Rightarrow soft\_drink$. First, we calculate the association using the conventional technique of conditional rules. We obtain $C(mustard \wedge sausage \wedge beer \Rightarrow chips) = 0.5$ and $C(mustard \wedge sausage \wedge beer \Rightarrow soft\_drink) = 0.5$. The results show that associations measured by the above two association rules are equal, indicating that pattern $mustard \wedge sausage \wedge beer$ is associated equally with pattern *chips* and with pattern $soft\_drink$. However, the results for the same two association rules are different if we use our new measure for association patterns. The results are $C(mustard \wedge sausage \wedge beer \Rightarrow chips) = -4/7$ and $C(mustard \wedge sausage \wedge beer \Rightarrow soft\_drink) = 1/3$. The above results show that pattern $mustard \wedge sausage \wedge beer$ is disassociated with pattern *chips*, but is associated with pattern $soft\_drink$, although the two patterns have the same conditional confidence. The above results mean that triplet pattern $mustard \wedge sausage \wedge beer$ is more likely to occur together with soft drink than without soft drink while the triplet pattern is less likely to occur together with chips than without chips. To see this difference more clearly, we modify the data in Table 1 by replacing the first 0 in column chips by 1 and the last two 1s in column Soft_drink by 0, so that association values are changed while conditional confidence values remain the same. However, the new association values become $C(mustard \wedge sausage \wedge beer \Rightarrow chips) = -3/4$ and $C(mustard \wedge sausage \wedge beer \Rightarrow soft\_drink) = 3/4$. The above results indicate that triplet pattern $mustard \wedge sausage \wedge beer$ is three times more likely to appear with soft drink than without it while the same triplet pattern is three times more likely to appear in the absence of chips than in the presence of chips. This difference between soft drink and chips in association with the same triplet pattern can not be seen from conditional rules.

Now, we apply our association rules to a more interesting case: data set for KDD Cup 98. Here, we try to identify interesting patterns associated with donation pattern defined by value 1 in data column TARGET_B. All conditional

Table 1. A simple example data set

Row_ID	Chips	Mustard	Sausage	Soft_drink	Beer
T_1	1	0	0	0	1
T_2	1	1	1	1	1
T_3	1	0	1	0	0
T_4	0	0	1	0	1
T_5	0	1	1	1	1
T_6	1	1	1	0	1
T_7	1	0	1	1	1
T_8	1	1	1	0	0
T_9	1	0	0	1	0
T_{10}	0	1	1	0	1

patterns discovered using conditional rules are shown in Table 2. Discovered conditional patterns in Table 2 are shown in two dimensions: support and association dimensions, using a new method developed for pattern-driven multidimensional pattern analysis (T. Zhang, US patent pending). A key for pattern-driven multidimensional analysis is to choose support and association measures as dimensional variables, and to display distribution of pattern numbers, not pattern

Table 2. Multidimensional distribution of conditional pattern numbers (SI - Support interval, CI - confidence interval)

SI				CI	×	10%				
SI	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
0-1%	25121	6680	2751	1382	136	1414	87	1	0	0
1-2%	2654	6	0	0	0	0	0	0	0	0
2-3%	1367	2	0	0	0	0	0	0	0	0
3-4%	782	2	0	0	0	0	0	0	0	0
4-5%	458	0	0	0	0	0	0	0	0	0
5-6%	326	0	0	0	0	0	0	0	0	0
6-7%	226	0	0	0	0	0	0	0	0	0
7-8%	162	0	0	0	0	0	0	0	0	0
8-9%	127	0	0	0	0	0	0	0	0	0
9-10%	102	0	0	0	0	0	0	0	0	0
10-20%	363	0	0	0	0	0	0	0	0	0
20-30%	68	0	0	0	0	0	0	0	0	0
30-40%	45	0	0	0	0	0	0	0	0	0
40-50%	34	0	0	0	0	0	0	0	0	0
50-60%	42	0	0	0	0	0	0	0	0	0
60-70%	23	0	0	0	0	0	0	0	0	0
70-80%	23	0	0	0	0	0	0	0	0	0
80-90%	42	0	0	0	0	0	0	0	0	0
90-100%	54	0	0	0	0	0	0	0	0	0

Table 3. Multidimensional distribution of association pattern numbers (SI - Support interval, AI - association interval)

SI				AI	×	10%				
SI	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
0-1%	2774	2954	2189	2572	2525	2466	2551	2929	2978	1669
1-2%	698	469	191	53	39	9	0	0	0	0
2-3%	413	213	85	28	13	7	1	0	0	0
3-4%	282	121	33	9	15	2	0	0	0	0
4-5%	177	73	9	0	10	0	0	0	0	0
5-6%	133	30	8	3	3	1	0	0	0	0
6-7%	98	22	3	1	1	0	0	0	0	0
7-8%	69	14	0	0	0	0	0	0	0	0
8-9%	56	12	1	1	1	0	0	0	0	0
9-10%	53	3	2	1	1	0	0	0	0	0
10-20%	168	27	9	1	1	0	0	0	0	0
20-30%	29	3	2	0	1	0	0	0	0	0
30-40%	17	2	0	0	0	0	0	0	0	0
40-50%	19	0	0	0	0	0	0	0	0	0
50-60%	13	0	1	0	0	0	0	0	0	0
60-70%	9	0	0	0	0	0	0	0	0	0
70-80%	4	0	0	0	0	0	0	0	0	0
80-90%	13	0	0	0	0	0	0	0	0	0
90-100%	14	0	0	0	0	0	0	0	0	0

details (this is a crucial difference from OLAP where aggregated data values are displayed in multiple dimensions), in multiple dimensions. The point here is to display distribution of interestingness of all discovered association rules in one screen shot before choosing thresholds for interesting patterns. Each two-dimensional lattice cell in Table 2 shows the number of patterns having support values within a given support interval and having association values within a given association interval. For example, the first cell in the first row shows 25121 association patterns having support values less than 1% and having association values less than 10%. Typically, the first row and first column represent noise because support values and/or association values are too small (1%, 10%). Then, the remaining cells represent association patterns that are not noise in the first cut. There are 10 such patterns in Table 2 measured by conditional rules. If we set thresholds for interesting patterns to be 5% for support and 40% for association (or conditional confidence). We found no interesting patterns in Table 2. The results for association patterns using our new method are shown in Table 3. First, we find many association patterns (in thousands) that are not noise patterns in the first cut. Most of these association patterns are difficult to separate from noise using conditional rules because they fall into the first column (or the second column if we count support interval column). There are two ways to discover association rules: user-driven and pattern-driven discoveries. In a user-driven discovery, a user specifies a pair of thresholds for interesting patterns. For

Table 4. Interesting association patterns

Field Names	Association Patterns	Support	Association
RFA_4	S4D $\Longrightarrow$ Donor	0.053479	0.508833
RFA_3	S4D $\Longrightarrow$ Donor	0.053273	0.499053
RFA_5	S4D $\Longrightarrow$ Donor	0.053066	0.493150
RFA_2	L4D $\Longrightarrow$ Donor	0.096015	0.488384
RFA_2A	D $\Longrightarrow$ Donor	0.143713	0.484016
RAMNT_14	5 $\Longrightarrow$ Donor	0.062771	0.446097
RAMNT_24	5 $\Longrightarrow$ Donor	0.056989	0.421484
LASTGIFT	5 $\Longrightarrow$ Donor	0.085071	0.411148
RFA_2F	4 $\Longrightarrow$ Donor	0.204419	0.410662

the same pair of thresholds: 5% for support and 40% for association, we found 9 interesting association patterns in Table 3. Detail of these 9 interesting association patterns and rules is shown in Table 4. In Table 4, column 1 shows data field names and column 2 shows field values. Furthermore, these 9 interesting patterns all have conditional confidence less than 10%, implying that they are in the first column in Table 2. Another way for discovery of association rules is to perform a pattern query by selecting a cell or a set of cells in Table 3. For example, select cell in row 6 (support values between 5% and 6%) and column 6 (association between 40% and 50%) and perform a pattern query. The resulting 3 interesting patterns are shown in rows 2, 3, and 7, in Table 4.

6 Verification of Association Rules

A rigorous test for the new association measure would be prediction of donors in the second example above using our new association rules. However, such tests require development of a solid prediction model for accurate predictions, which involves intelligent binning among many other issues to be resolved accurately. Instead of developing a sophisticated prediction model, we performed a simple test to verify the accuracy of our new association rules. In our test, we separate data records in two groups: records associated with donors and records disassociated with donors in the KDD Cup 98 example above. We measure patterns associated with donors. Patterns having association values larger than zero are association patterns while patterns having association values less than zero are disassociation patterns. Assume a donor is associated with a set of patterns having an average association to be no less than zero. We can verify such association by the following consideration. Calculate the average association for each record by averaging association values from various columns (except TARGET_B and TARGET_D columns) in each record. Assign each record with its average association. In theory, records having higher average association would be more likely associated with donors. Ideally, all records having average association larger than zero are associated with donors. We tested this simple model on the KDD Cup 98 learning data set. The results are shown in Table 5. In Table 5,

Table 5. Validation of association rules

Filter	Record range	Average association	Num. of records	Num. of donors
NO	1 – 95412	$C < 0$	68866	1151
NO	1 – 95412	$C \geq 0$	26546	3692
$[-0.1, 0.1]$	1 – 95412	$C < 0$	66602	921
$[-0.1, 0.1]$	1 – 95412	$C \geq 0$	28810	3922
$[-0.3, 0.3]$	1 – 95412	$C < 0$	74508	583
$[-0.3, 0.3]$	1 – 95412	$C \geq 0$	20904	4260
$[-0.5, 0.5]$	1 – 95412	$C < 0$	82277	121
$[-0.5, 0.5]$	1 – 95412	$C \geq 0$	13135	4722
$[-0.7, 0.7]$	1 – 95412	$C < 0$	84686	46
$[-0.7, 0.7]$	1 – 95412	$C \geq 0$	10726	4797
$[-0.9, 0.9]$	1 – 95412	$C < 0$	85414	1
$[-0.9, 0.9]$	1 – 95412	$C \geq 0$	9998	4842
$[-0.98, 0.98]$	1 – 95412	$C < 0$	85466	0
$[-0.98, 0.98]$	1 – 95412	$C \geq 0$	9946	4843

the first column shows thresholds for association values to be considered in the calculation of average association per record. We filter out association values within a pair of thresholds in the calculation. For example, exclude association values between -0.1 and +0.1 in the calculation. The second column shows a data range in records we tried for each test on the learning data set (95412 records in total). In each data range, we divide records into two groups differing in the sign of average association per record. One group has average association no less than zero and the rest is the other group. The third column in Table 5 shows average association either no less than zero defining donor group or less than zero for non-donor group. Column 4 shows the number of records in each group. Column 5 shows the number of donors in each group. For example, the first two rows show that 68866 records have average association less than zero. 1151 of them are associated with a donor. 26546 records in the same test have non-negative average association and 3692 of them are associated with donors. It is seen that the results become better and better as the threshold defining a range to be filtered out becomes larger and larger. Our new association rules can be used for building a prediction model. For example, the above model for verification of association rules is the simplest prediction model using association rules, in which potential donors are predicted by positive average association per record. However, development of a solid prediction model using association rules requires solving other complicated problems such as measure of multiple association and intelligent binning, which is beyond what is intended for this paper.

7 Summary

We have presented new association rules with theoretical comparison with two leading existing measures. Our new measure of association is a transactional measure sensitive to association direction. We shown that the new association rules passed successfully a test of 3 well-known associations. In contrast, association rules using conditional measure or the χ^2 test were shown to fail the test. Comparison with conditional rules was shown experimentally. A new technique for supervised mining of association rules was presented in a pattern-driven way and in multiple dimensions. Both association rules and pattern-driven multidimensional pattern analysis presented here are implemented in Triada's products using a new pattern-base technology (Bugajski and Russo 1993, Bugajski and Russo 1994, Bugajski 1997, Zhang 1999 and Zhang et al. 1999). Finally, we verified our new association rules experimentally by grouping records into association class and disassociation class for a given target pattern. We shown that the target pattern appears only in the association group if a classifier is properly chosen. However, we point out that association rules presented here measure single association only. Association rules for measure of multiple-association relationships are more complicated and will be presented elsewhere. In conclusion, we believe that we have solved a fundamental problem in understanding association rules by presenting the first correct association rules.

References

R. Agrawal, T. Imielinski, and A. Swami. 1993a. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 207-216.

R. Agrawal, T. Imielinski, and A. Swami. 1993b. Database mining: a performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5:914-925.

R. Agrawal and R. Srikant. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487-499.

R. Agrawal, A. Arning, T. Bollinger, M. Mehta, J. Shafer, and R. Srikant. 1996a. The Quest Data Mining System. In *Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data.*

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. 1996b. Fast Discovery of Association Rules. In, Fayyad et al (Fayyad et al., 1996), pages 307-328.

J. M. Bugajski and J. T. Russo. 1993. US patent No. 5,245,337.

J. M. Bugajski and J. T. Russo. 1994. US patent No. 5,293,164.

J. M. Bugajski. 1997. US patent No. 5,592,667.

A. W. Cheung, J. Han, V. T. Ng, A. Fu, and Y. Fu. 1996. A fast distributed algorithm for mining association rules. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems.*

U. M. Fayyad. G. Piatetsky-Shapiro, P. Smyth, and R. Uthrusamy. 1996. *Advances in Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA.

J. Han and Y. Fu. 1995. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 420-431.

M. Houtsman and A. Swami. 1995. Set-oriented mining of association rules. In *Proceedings of the International Conference on Data Engineering*, pages 25-34.

M. Klemettine, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. 1994. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the 3th International Conference on Information and Knowledge Management*, pages 401-407.

S. D. Lee, D. W. Cheung, and B. Kao. 1998. Is sampling useful in data mining? A case in the maintenance of discovered association rules. Data Mining and Knowledge Discovery, Vol. 2, number 3, pages 233-262.

H. Mannila, H. Toivonen, and A. I. Verkamo. 1994. Efficient algorithms for discovering association rules. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, pages 144-155.

H. Mannila. 1998. Database methods for data mining. Tutorial for *4th International Conference on Knowledge Discovery and Data Mining.*

R. Meo, G. Psaila, and S. Ceri. 1998. An extension to SQL for mining association rules. Data Mining and Knowledge Discovery, Vol. 2, number 2, pages 195-224.

J. S. Park, M. S. Chen, and P. S. Yu. 1995. An effective hash based algorithm for mining association rules. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 175-186.

A. Savasere, E. Omiecinski, and S. Navathe. 1995. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 432-444.

C. Silverstein, S. Brin, and R. Motwani. 1998. Beyond market basket: Generalizing association rules to dependence rules. Data Mining and Knowledge Discovery, Vol. 2, number 1, pages 39-68.

R. Srikant and R. Agrawal. 1995. Mining generalized association rules. In *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 407-419.

H. Toivonen. 1996. Sampling large databases for finding association rules. In *Proceedings of the 22th International Conference on Very Large Data Bases*, pages 134-145.

J. Wijsen and R. Meersman. 1998. On the complexity of mining quantitative association rules. Data Mining and Knowledge Discovery, Vol. 2, number 3, pages 263-282.

T. Zhang, J. M. Bugajski, and K. R. Raghavan. 1999. US patent No. 5,966,709.

T. Zhang. 1999. US patent No. 5,983,232.

Density-Based Mining of Quantitative Association Rules

David W.Cheung, Lian Wang, S.M.Yiu, and Bo Zhou

Department of Computer Science and Information Systems
The University of Hong Kong, Pokfulam, Hong Kong
{dcheung,wlian,smyiu,bzhou}@csis.hku.hk

Abstract. Many algorithms have been proposed for mining of boolean association rules. However, very little work has been done in mining quantitative association rules. Although we can transform quantitative attributes into boolean attributes, this approach is not effective and is difficult to scale up for high dimensional case and also may result in many imprecise association rules. Newly designed algorithms for quantitative association rules still are persecuted by nonscalable and noise problem. In this paper, an efficient algorithm, QAR-miner, is proposed. By using the notion of "density" to capture the characteristics of quantitative attributes and an efficient procedure to locate the "dense regions", QAR-miner not only can solve the problems of previous approaches, but also can scale up well for high dimensional case. Evaluations on QAR-miner have been performed using both synthetic and real databases. Preliminary results show that QAR-miner is effective and can scale up quite linearly with the increasing number of attributes.

1 Introduction

Data mining, the effective discovery of correlations among the underlying data in large databases, has been recognized as an important area for database research and has also attracted a lot of attention from the industry as it has many applications in marketing, financial, and retail sectors. One commonly used representation to describe these correlations is called *association rules* as introduced in [3]. In this model, the set $I = \{i_1, i_2, ..., i_m\}$ is a collection of items or attributes. The database DB consists of a set of transactions, where each transaction is a subset of items in I. An association rule is an implication of the form $X \Rightarrow Y$ with $X, Y \subseteq I$ and $X \cap Y = \emptyset$. The meaning of the rule is that a transaction contains items in X will likely contains items in Y so that marketing strategy can be derived from this implication, for instance. To determine whether an association rule is interesting, two thresholds are being used: *support* and *confidence*. An association rule, $X \Rightarrow Y$, has support $s\%$ in DB if $s\%$ of transactions in DB contain items in $X \cup Y$. The same association rule is said to have confidence $c\%$ if among the transactions containing items in X, there are $c\%$ of them containing also items in Y. So, the problem is to find all association rules which satisfy pre-defined minimum support and minimum confidence constraints.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 257–268, 2000.
© Springer-Verlag Berlin Heidelberg 2000

In this setting, attributes which represent the items are assumed to have only two values and thus are referred as *boolean* attributes. If an item is contained in a transaction, the corresponding attribute value will be 1, otherwise the value will be 0. Many interesting and efficient algorithms have been proposed for mining association rules for these boolean attributes, for example, Apriori [3], DHP [9], and PARTITION algorithms [10] (see also [1,2,12,4,5,7]). However, in a real database, attributes can be *quantitative* and the corresponding domains can have multiple values or a continuous range of values, for example, Age, Salary. By considering this type of attributes, association rules like this one, $(30 \leq \text{Age} \leq 39)$ and $(50000 \leq \text{Salary} \leq 79999) \Rightarrow (100000 \leq \text{Loan} \leq 300000)$, will be desirable. To handle these quantitative attributes, in this paper, a new threshold called *density* will be introduced. This new threshold, together with the support and confidence thresholds, will lead to an efficient and scalable algorithm, QAR-miner, for mining quantitative association rules.

1.1 Motivation for a Density Threshold

The motivation for a new *density* threshold can best be illustrated by an example. Assuming that we have two quantitative attributes, A and B (see figure 1). Each transaction in the database is mapped to a data point (or a range) in this two dimensional space using the corresponding values of the attributes as coordinates. We want to find all the association rules of the form $A \subseteq [x_1, x_2] \Rightarrow B \subseteq [y_1, y_2]$ where $x_1, x_2 \in \{0, a1, a2, a3, a4, a5, a6\}$ with $x_2 > x_1$ and $y_1, y_2 \in \{0, b1, b2, b3, b4, b5\}$ with $y_2 > y_1$. And we further set the support threshold to 5 points and the confidence threshold to 50%. We can obviously obtain the

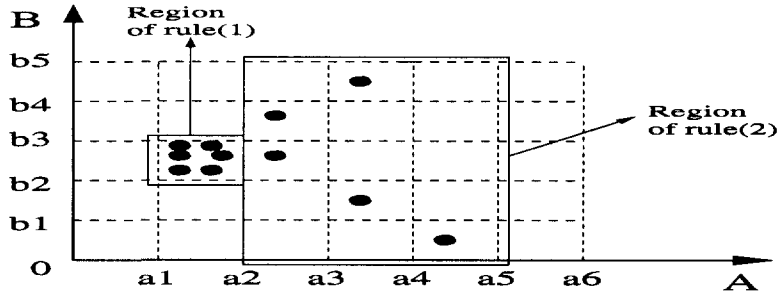


Fig. 1. Example of some quantitative rules

following rules:

- $A \subseteq [a1, a2] \Rightarrow B \subseteq [b2, b3]$ (1)
- $A \subseteq [a2, a5] \Rightarrow B \subseteq [0, b5]$ (2)
- $A \subseteq [0, a5] \Rightarrow B \subseteq [0, b5]$ (3)

One can easily see that with only the support and confidence thresholds, as long as a range has the minimum support, any larger range containing this range

will also satisfy the support threshold. Similarly, by enlarging the two ranges in both dimensions, it is very *likely* that the new ranges will satisfy the confidence constraint as well. This can lead to many "useless" rules:

- **Trivial Rule:** Rules (2) and (3) will be considered as useless and not interesting because rule (3) covers all possible values of both attributes while rule (2) covers almost all possible values of both attributes.
- **Redundant Rule:** According to the above observation, from rule (1), we can have all kinds of rules in the form $A \subseteq [z1, z2] \Rightarrow B \subseteq [u1, u2]$ where $[a1, a2] \subseteq [z1, z2]$ and $[b2, b3] \subseteq [u1, u2]$. In this particular example, we can see that all these rules satisfy both the support and confidence constraints. However, these rules, when compared with rule (1), are not very interesting because the increase in support of these rules is relatively a lot less than the increase in the sizes of the ranges.

From the above example, intuitively, rule (1) is much more preferable than both rules (2) and (3). The reason is the density of the region representing rule(1) is much higher than the density of regions representing rule(2) and (3)(see Fig. 1). Hence, if density is defined as a new threshold, it is easy to get rid of trivial and redundant rules.

In real application, when we map a database in a multidimensional space, we can always notice that the data points (transactions) exhibit a "dense-regions-in-sparse-regions" property. In other words, the space is sparse but not uniformly so. That is, the data points are not distributed evenly throughout the multidimensional space. According to this kind of distribution and the density threshold we have just introduced, the problem of mining quantitative association rules can be transformed to the problem of finding regions with enough density (*dense regions*) and finally these dense regions will then be mapped to quantitative association rules.

1.2 Related Work

There are a few others' work in trying to solve this mining problem for quantitative attributes. In [11], the authors proposed an algorithm which is an adaptation of the Apriori algorithm for quantitative attributes. It partitions each quantitative attribute into consecutive intervals using *equi-depth* bins. Then adjacent intervals may be combined to form new intervals in a controlled manner. From these intervals, *frequent itemsets* (c.f. *large itemsets* in Apriori Algorithm) will then be identified. Association rules will be generated accordingly. The problems with this approach is that the number of possible interval combinations grows exponentially as the number of quantitative attributes increases, so it is not easy to extend the algorithm to higher dimensional case. Besides, the set of rules generated may consist of redundant rules for which they present a "greater-than-expected-value" interest measure to identify the interesting ones.

Another algorithm proposed for quantitative attributes is [8]. Their idea is to combine similiar association rules to form interesting quantitative association rules using the technique of clustering. The algorithm will map the whole

database into a two dimensional array with each entry representing an interval in each of the dimensions. Entries with enough support will be marked, then a greedy clustering algorithm using "bitwise AND" operation is used to locate the clusters. In this approach, the drawback is that the algorithm is sensitive to noise. Although an image processing technique, called *low-pass filter*, is used to remove these noises, the algorithm is still sensitive to noise and noise is unavoidable in real database. Also, the algorithm is basically designed for two quantitative attributes, so again it is not trivial to extend the algorithm to an efficient one for higher dimensional cases.

On the other hand, the noise problem and the redundant rules problem of these approaches can be handled by the density threshold in our approach. And our QAR-miner can be used in higher dimensional cases with scalable performance. It is hoped that this new approach can give more insights on this mining problem. The remaining of the paper will be organized as follows. Some preliminary definitions will be given in section 2. Section 3 will describe the algorithm for QAR-miner. Evaluations on QAR-miner will be discussed in section 4. Conclusion will be presented in section 5.

2 Some Preliminary Definitions

In this section, some preliminary notations and definitions will be presented.

Definition 1. *Let A_i $(1 \leq i \leq n)$ be a quantitative attribute with a totally ordered domain. Then, a quantitative association rule is of the form:*

$A_1 \subseteq [a_1, b_1] \wedge A_2 \subseteq [a_2, b_2] \wedge ... \wedge A_{n-1} \subseteq [a_{n-1}, b_{n-1}] \Rightarrow A_n \subseteq [a_n, b_n]$ *where* $[a_i, b_i]$ $(a_i \leq b_i)$ *is a range on the domain of* A_i.

The mapping of transactions to data points of a multidimensional space is done as follows. Each attribute is represented by a dimension in the space. A transaction is mapped to a data point in the space using the attribute values as coordinates. And in the multi-dimensional space, the volume and the density of a rectangular region r will be denoted by v_r and ρ_r respectively.

Definition 2. *Consider a multidimensional space S created by a mapping from a set of transactions. Let r be a rectangular region in S, then the density of r, ρ_r, is defined as the number of data points inside the region divided by v_r. (We sometimes use $\rho(r)$ to denote ρ_r for simplicity.)*

Definition 3. *Let X_i $(1 \leq i \leq q)$ denote a set of regions. Then, if X is the region formed by combining this set of regions, X is the minimum bounding rectangular box which contains all regions X_i.*

Definition 4. *S is a* **dense region** *if its density, ρ_S, is greater than or equal to a user specified threshold, ρ_{min}.*

Definition 5. *Given n parts A_1,A_2,...,A_n, which are n sets of disjoint regions. The* **maximum combined density** *of these n parts is*

$$max\{\rho_C | C \text{ is the combined region of } c_1, c_2, ..., c_n \text{ and } c_1 \subseteq A_1, c_2 \subseteq A_2, ..., c_n \subseteq A_n\}$$

3 Algorithms for QAR-Miner

3.1 Framework of QAR-Miner

In QAR-miner, the multidimensional space is divided into *cells*. The problem of mining quantitative association rule is divided into two subproblems: (1) finding dense regions and (2) translating dense regions to quantitative association rules. We further decompose the first step into two subproblems. We first locate all *base dense regions*. A base dense region is a connected set of cells in which each cell must have a density higher than ρ_{low}. Then, these base dense regions will be combined to form bigger dense regions. The formation of these bigger dense regions is to combine similar quantitative rules into one interesting rule. However, in this part, if at each step, we try all possible combinations of the available dense regions, the complexity will be extremely high. To reduce the time complexity, a "hierarchical divide-and-conquer" approach is developed. Finally, the dense regions identified in step (1) will be tested against the support and confidence thresholds to generate quantitative association rules.

3.2 EDEM Algorithm for Finding Base Dense Regions

We formulate the base dense region discovering problem as the following optimization problem. Let $S = D_1 \times D_2 \times \cdots \times D_d$ be a d-dimension space such that, for $1 \leq i \leq d$, $D_i = \{x | x \in A_i, L_i \leq x < H_i\}$ is a range in a totally ordered domain A_i, bounded above and below by H_i and L_i respectively. The space S is partitioned into equal size cells such that the *cell length* on the i-th dimension is c_i. That is, the i-th dimension is divided into $cn_i = (H_i - L_i)/c_i$ equal intervals. We use $CP =< c_1, c_2, \cdots, c_d >$ to denote a *cell-based partition* of S. We use cell as the basic unit to reference the coordinates of regions. We use $r = [(l_1, l_2, \cdots, l_d), (h_1, h_2, \cdots, h_d)]$ to denote a region representing a subspace whose projection on the i-dimension is the interval $[L_i + c_i l_i, L_i + c_i h_i]$. And ρ_{min} is the density requirement for base dense regions. Because the final rules we mined should satisfy the support threshold, so the volume of a base dense region corresponding to a rule should also be large enough or it cannot get enough support. Hence we give v_{min} as the volume threshold that a base dense region should satisfy. ρ_{low} is another density threshold that each cell should satisfy in a base dense region. This is because we do not want to combine empty or nearly empty cells in a base dense region.

Given a d-dimensional space S, a set of data points D in S, a cell based partition CP on S, together with three input thresholds ρ_{min}, ρ_{low}, and v_{min}, finding the base dense regions in S is equivalent to solving the following optimization problem (see table 1 where r_i denotes the ith base dense region).

The EDEM algorithm actually consists of three steps:

- Step 1: Build a k-d tree to store the non-empty cells.
- Step 2: Prune away sparse regions from the k-d tree.
- Step 3: Locate the base dense regions.

Table 1. Problem statement of finding base dense regions

Objective:	Maximize $\Sigma v_{r_i} (i = 1, ...n)$
Constraints:	$\rho_{r_i} \geq \rho_{min}, (i = 1, ..., n)$
	$v_{r_i} \geq v_{min}, (i = 1, ..., n)$
	$\forall i = 1, ..., m$ for all cell cl in r_i, $\rho_{cl} \geq \rho_{low}, (i = 1, ..., n)$

In fact, after step 2, most of the sparse regions will be pruned away. The k-d tree will be divided into covers containing the base dense regions. It is guaranteed that the same base dense region will not be divided into different covers although more than one base dense region may be in the same cover. Therefore, in step 3, the searching can be done in each cover. Based on some interesting theorems, these three steps can be performed efficiently. Please refer to [13] for more details on how this is done.

3.3 HDRCluster Algorithm for Combining Base Dense Regions

Now what we have is a set of base dense regions. The next step is to combine them into bigger dense regions. The idea is that: Given a set of dense regions, by combining these regions in all possible cases, the combination with the largest combined density which still satisfy the density constraint should be combined. The final set of dense regions so obtained is regarded as the optimal result. However, a direct implementation of this idea is highly inefficient. Fortunately, based on the following observations, a more efficient algorithm, HDRCluster, can be used.

Let us illustrate the observations by an example. In this example, R, which is the set of input base dense regions, contains 15 base dense regions which are labelled with integers from 1 to 15. For presentation purpose, let DRCluster(k) denote the procedure that will try all combinations of regions with at most k regions in any combination. The results generated by DRCluster(3) and DRCluster(2) are shown in Figure 2 (a) and (b), respectively. In these figures, rectangles surrounded by dashed line denote the dense regions generated by the processes.

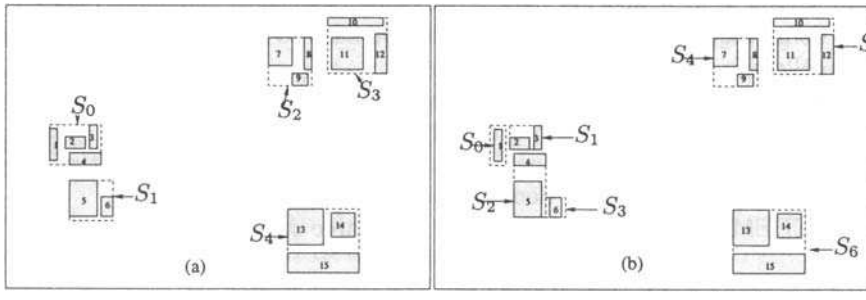


Fig. 2. Results of DRCluster (3) and DRCluster(2) on R

After comparing these two figures, we have the following two observations. Firstly, although DRCluster(2) does not generate the same result as DRCluster(3), it does generate some dense regions generated by DRCluster(3). Secondly, some base dense regions are "far" away, some combinations of regions can be ignored without affecting the final result. To formalize the second observation, we define *independent parts* as follows:

Definition 6. *Given n parts such that each part is a set of dense regions. If the maximum possible combined density of any combination of these parts is less than ρ_{min} These parts are* **independent parts.**

And with the help of the following theorem, it is easy to identify independent parts.

Theorem 1. *Given that DRCluster(p) can generate the optimal result. And P_1,...,P_n are n parts of base dense regions. If the maximum combined density of any two parts is smaller than $\frac{2*\rho_{min}}{min\{p,n\}}$, they are independent parts.*

These two observations together give a hierarchical divid-and-conquer algorithm for combining dense regions. See below for a high level description of the algorithm and the detailed version will be presented in the full paper.

- Step 1: Set $k = 2$.
- Step 2: Divide the input R into independent parts.
- Step 3: For each independent part, run DRCluster(k).
- Step 4: Remove regions which are already optimal.
- Step 5: Set $k = k + 1$. Repeat steps 2 - 5 until all optimal regions are identified.

3.4 Generate Quantitative Association Rules

Now we can transform dense regions to quantitative association rules. Since density threshold is already satisfied, so we only need to consider whether support and confidence threshold are satisfied. From the definition, a quantitative association is of the form: $A_1 \subseteq [a_1, b_1] \wedge A_2 \subseteq [a_2, b_2] \wedge ... \wedge A_{n-1} \subseteq [a_{n-1}, b_{n-1}] \Rightarrow A_n \subseteq [a_n, b_n]$ where $A_i (1 \leq i \leq n)$ is a quantitative attribute. It is obvious that the rule defines a dense region by the ranges for each A_i. Let this dense region be denoted by R_n. And let the orthographic projection of R_n on the hyperplane formed by dimensions $A_1, A_2, ..., A_{n-1}$ be denoted by R_{n-1}. Because of orthographic projection, the length of R_{n-1} on ith dimension is the same as it of R_n, where $1 \leq i \leq n-1$. Now the support of this potential association rule that generated from R_n is the number of points falling in R_n, and the confidence of this potential rule is the number of points falling in R_n over the number of points falling in R_{n-1}. If both of the support and confidence requirements are satisfied, then a quantitative association rules is successfully generated from the dense region R_n. See an example in Figure 3. We can see a three dimensional box R_n in the three dimensional space, and a two dimensional shadow below it

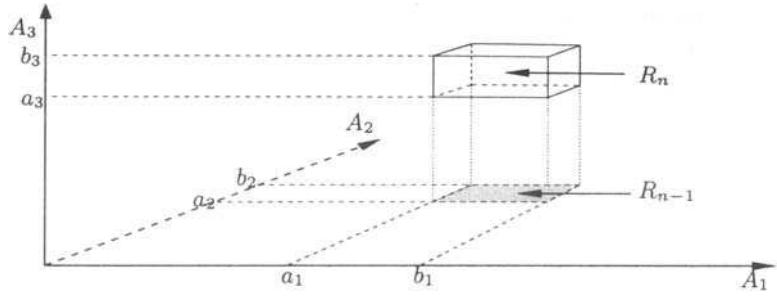


Fig. 3. Example of counting support and confidence

which is R_{n-1}, that is, the orthographic projection of R_n on the plane formed by dimensions A_1 and A_2.

As we perform the same checking on all the dense regions, we can get all the quantitative association rules.

4 Performance Study of QAR-Miner

Some experiments have been carried out to evaluate the performance of QAR-miner. All the experiments are performed on a Sun Sparc 5 workstation running Solaris 2.6 with 64M main memory. Since the third step of QAR-miner is rather trivial, so the discussion will be mainly focused on the first two steps.

4.1 Generation of Synthetic Database

In this performance study, we first use synthetic database to evaluate the performance of QAR-miner. The main parameters for synthetic database generation are listed in Table 2. The databases are generated by a 2-step procedure. In the first step of the procedure, a number of non-overlapping potential dense regions are generated. In the second step, points are generated in each of the potential dense regions, as well as in the remaining space. For each generated point, transaction(s) corresponding to that point will then be generated.

4.2 Evaluation of EDEM and HDRCluster on Synthetic Database

To simplify the experiments, we use a default cell volume of 20 for the cells in EDEM. And we set $\rho_{low} = \rho_{min}/2$ and $v_{min} = 4096$ and $\rho_{min} = \overline{\rho}_{dr}$, in all the experiments.

Effect of EDEM and HDRCluster on Different Dimension Numbers. In these set of experiments, we fixed the volume of the d-dimensional space and increased the number of dimensions from 2 to 7. The d-dimensional space has a volume of 2×10^{10} with different lengths in dimensions. Also, $N_{dr} = 10$,

Table 2. Input parameters of data generation

parameter	meaning
d	no. of dimensions
L_i	length of dimension i
ρ_s	density of the sparse region
m	average multiplicity for the whole space
N_{dr}	no. of dense regions
$\overline{l}_i$	average length of dense regions in dimension i
σ_i	standard deviation of the length of dense regions in dimension i
$\overline{\rho}_{dr}$	average density of dense regions
$\overline{m}_{dr}$	average multiplicity for the dense regions

$\overline{\rho}_{dr}$=20%. The average volume of a potential dense region is 5×10^5, and the number of data points in the whole d-dimensional space is about one million, in which about 5% are sparse points.

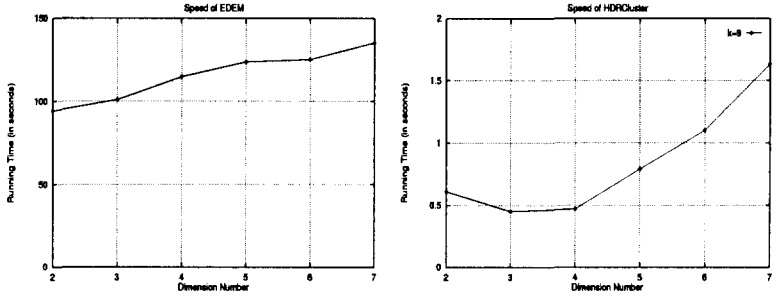


Fig. 4. Speed of EDEM and HDRCluster on different dimension numbers

Figure 4 clearly shows that the speeds of EDEM and HDRCluster are not increasing exponentially as the number of dimensions increases. This is what we have expected from the analysis.

Effect of EDEM and HDRCluster on Different Number of Dense Regions. Besides the above experiments, we also test the performance of EDEM and HDRCluster with different number of dense regions from 10 to 100. These experiments are performed in a 3-dimensional space with a volume of 2×10^{10}, and the total number of points is about one million in which about 5% of them are sparse points. Figure 5 shows the result.

In Figure 5, we can see that the speed of EDEM decreases as the number of dense regions increases. The reasons are the followings. Firstly, the size of each dense region becomes smaller. This reduces the amount of splitting of the base dense regions across the k-d tree nodes; and secondly as the size of dense region

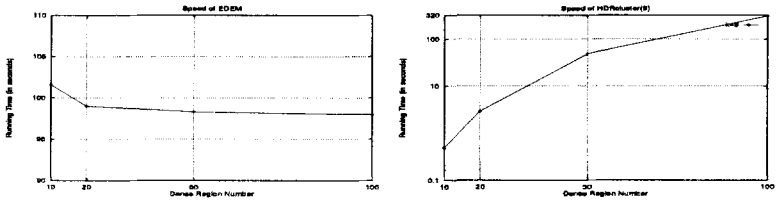


Fig. 5. Speed of EDEM and HDRCluster with different number of dense regions

decreases, the size of base dense region covers becomes smaller accordingly. This in turn speeds up the base dense region growing in these base dense region covers. We may also notice that the speed of HDRCluster is growing rapidly with the number of base dense regions. So the smaller the value of k in HDRCluster(k), the better the performance.

In the above experiments, we also test the results generated by HDRCluster(k) with different values of k ($k = 9, 8, 7, 6, 5, 4, 3$), and it is found that even with $k = 4$, the procedure will return the optimal result. We also counted the average and maximum number of base dense regions in a dense region when the number of dense regions changing form 10 to 100 in above experiments. Table 3 shows the corresponding result, it is clear that the average number of base dense regions in a dense region is always around 5, and the maximum number of base dense regions is seldomly exceed 8, only once. This is in line with our prediction. That is using a smaller value of k for the procedure HRDCluster(k) is usually good enough to get a good result.

Table 3. Average and Maximum base dense regions in dense regions

No of Dense Region	10	20	40	60	80	100
Average No	6.3	5.8	5.2	5.1	4.6	4.2
Maximum No	10	8	8	7	6	6

To further verify that the performance of HDRCluster(k) is better than that of DRCluster(k), the following figure compares the speeds of DRCluster(3), DRCluster(4), and HDRCluster(4). It is clear HDRCluster(4) outperforms the other two.

4.3 Evaluation of QAR-Miner on Real Database

We also test QAR-miner on real data. The real data is about the **world wide re-export trade statistics** provied by Hong Kong Productivity Council.

In this set of real data, we have chosen four attributes(dimensions) from the original database for our experiment: Country of Consignment(denoted by CC), Country of Origin (denoted by CO), Trade Item(denoted by TI) and Month. Each tuple in the database represents that Hong Kong has re-exported (a trade

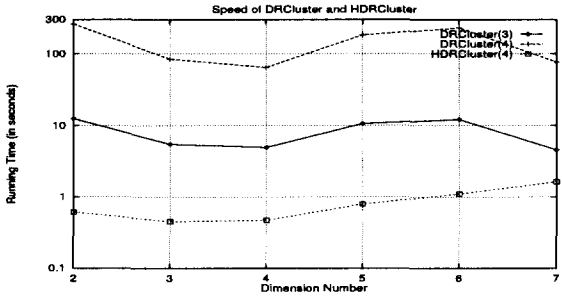


Fig. 6. Speed of DRCluster and HDRCluster with different number of dimension

item in) TI provied by (a country in)CO to (a country in) CC in that month. Both CO and CC have 188 different values with each value representing a country or a region. TI has 6343 different values with each value representing a trade item. The rule is of the form

$$[a, b] \subseteq CO \wedge [c, d] \subseteq CC \wedge [i, j] \subseteq Month \Rightarrow [e, f] \subseteq TI$$

([a,b] and [c,d] are set of countries, [e,f] is a set of trade items.) The number of tuples is about 1,200,000 and the volume of this 4-dimensional space is about 1.5×10^9. Here we set ρ_{min}=30%, and $v_{min} = 2000$ After running QAR-miner, we find some interesting rules, the following shows one of them (the dots in rules means there are other items that we do not list them here):

$$\left\{\begin{matrix}\text{PRC}\\ \text{JAPAN}\end{matrix}\right\} \bigwedge \left\{\begin{matrix}\text{FINLAND,CHILE}\\ \text{GREECE,TURKEY}\\ \text{MEXICO}\end{matrix}\right\} \bigwedge \{5\text{-}9\} \Rightarrow \left\{\begin{matrix}\text{BALL POINT PENS}\\ \text{MEDICAMENTS}\\ \text{FOOTWEAR...}\end{matrix}\right\}$$

5 Conclusion

In this paper, we have introduced a new "density" threshold for mining quantitative association rules. Using this density threshold and an efficient algorithm for locating dense regions, an efficient algorithm, QAR-miner, is developed for quantitative attributes. This QAR-miner not only solves the problems of previous algorithms, but also can scale up well for high dimensional cases as supported by the preliminary experimental results.

In fact, the techniques presented in this paper can also be applied in other areas. For example, both EDEM and the HDRCluster algorithms are found very useful in indexing OLAP data for reducing the query response time [6].

References

1. R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proc. 4th Conf. on Foundations of Data organization and Algorithms*, October 1993.
2. R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifier for database mining applications. In *Proc. of the 18th Conf. VLDB*, Vancouver, Canada, August 1992.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. Of the 20th Conf. on VLDB*, Santiago, Chile, September 1994
4. D. W. Cheung, V. T. NG, A. W. Fu, and Y.J. Fu. Efficient mining of association rules in distributed databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(6)1996
5. D.W. Cheung and Y. Xiao. Effect of Data Skewness in Parallel Mining of Association Rules. In *Proc. The 2th PAKDD-98 Conf.* . Melbourne, Australia, April, 1998.
6. David W. Cheung et al. Towards the Building of a Dense-Region-Based OLAP System. *Manuscript*, 1999.
7. J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. Of the 21st Conf. on VLDB*, Zurich, Switzerland, September 1995.
8. B. Lent. A. Swami, and J. Widom. Clustering Association Rules. In *Proceedings of ICDE97*, Birmingham, UK, April 1997.
9. Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. ACM SIGMOD International Conference on management of Data*, San Jose, California, May 1995.
10. A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21th conf. VLDB*, Zurich, Switzerland, September. 1995
11. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. Of the ACM SIGMOD Conference on Management of Data*, Montreal, Canada, June 1996
12. H. Toivonen. Sampling large databases for association rules. In *Proc. Of the 22th Conf. on VLDB*, Mumbai, India, September 1996
13. B. Zhou, D.W. Cheung and B. Kao. A Fast Algorithm for Density-Based Clustering in Large Database. In *Proc. The 3rd (PAKDD-99) Conf.*, Beijing, China, April 1999.

AViz: A Visualization System for Discovering Numeric Association Rules

Jianchao Han and Nick Cercone

Department of Computer Science, University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
{j2han@hopper,ncercone@math}.uwaterloo.ca

Abstract. We introduce an interactive visualization system, AViz, for discovering numerical association rules from large data sets. The process of interactive visual discovery consists of six steps: preparing the raw data, visualizing the original data, cleaning the data, discretizing numerical attributes, and discovering and visualizing association rules. The basic framework of the AViz system is presented and three approaches to discretize numerical attributes, including equal-sized, bin-packing based equal-depth, and interaction-based approaches, are proposed and implemented. The algorithm for discovering and visualizing numerical association rules is discussed and analyzed. The AViz system has been experimented on a census data set. The experimental results demonstrate that the AViz system is useful and helpful for discovering and visualizing numerical association rules.

Keywords: KDD, data mining, data visualization, association rules.

1 Introduction

Many techniques and systems for data visualization have been developed and implemented [3,5,6,7,8,10]. One common feature of these business systems is their dependence on computer graphics and scientific visualization; data mining visualization is treated in a straightforward way to mine data so that the complex data can be made more understandable. The problem that exists in these systems, however, is that in most cases, the complex data is carefully arranged and displayed in specific visual form, and the mining results are left to the user who must observe and determine the meaning of the pictures. Unfortunately, it is not easy for a user to do this job because it usually requires a wealth of background knowledge. Silicon Graphics developed a series of visualizers like Map Visualizer, Tree Visualizer, etc. [5] to visualize data mining results according to different data mining techniques such as decision tree, neural network, etc. But only the mining results are displayed. Interactive visual data mining should provide a user with not only the mining results but also the entire process in visual form so that the user can participate in the mining process and present what he/she is concerned with most.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 269–280, 2000.
© Springer-Verlag Berlin Heidelberg 2000

We introduce an interactive system for visualizing the process of discovering numerical association rules. Consider association rules of the form: $A \Rightarrow B$, where A consists of two different numerical attributes and B is a numerical or nominal attribute. Suppose X and Y are two such different numerical attributes and Z is a quantitative attribute. Our goal is to find an interest region in the $X \times Y$ plane for each value z of Z, as shown below:

$$X \in [x_1, x_2], Y \in [y_1, y_2] \Longrightarrow Z = z$$

For example,

$$age \in [26, 35], salary \in [50k, 65k] \Longrightarrow position = analyst$$

is such a rule, where X is *age*, Y is *salary*, and Z is *position*.

The AViz system consists of six components, including data preparation, raw data visualization, data reduction, numerical attribute discretization, discretization visualization, and discovery and visualization of rules. In Section 2, related research is introduced. The framework of the AViz system is presented in Section 3, and three approaches to discretizing numerical attributes are discussed and compared in Section 4. The paradigm and algorithm for discovering and visualizing numerical association rules based on the scheme proposed by Fukuda et al. [4] are described and analyzed in Section 5. The implementation of the AViz system and an experiment on census data are discussed in Section 6. Finally, Section 7 contains concluding remarks.

2 Related Work

The main idea of discovering numerical association rules is to discretize the numerical attributes into disjoint intervals, and then transform the problem to mining item association rules [1,2,13]. Each interval is represented as a *Boolean* attribute. A tuple is said to satisfy this *Boolean* attribute if it has the value of the corresponding numerical attribute falling into this interval. There are currently many approaches to discretizing numerical attributes. The *equi-sized approach* is to simply partition the continuous domain into intervals with equal length [4]. The *equi-depth approach* [12] basically partitions the data values into intervals with equal size along the ordering of the data. Another equi-depth approach proposed in [13] is based on the measure of the partial completeness over itemsets which compensates for the amount of information lost by partitioning. The *distance-based approach* [11] consists of two phases, and addresses the measure of the quality of an interval and the distance between the data points in the adjacent intervals. The first phase identifies data clusters and the second phase combines clusters to form rules,

The AViz system provides three approaches to perform discretization, equi-sized, bin-packing based equi-depth, and interaction-based. The equi-sized and bin-packing based equi-depth approaches require the user to specify the number of intervals for both numerical attributes. The interaction-based approach is

based on one of the other two approaches. After the numerical attributes are discretized and visualized, the user can intuitively adjust the partition by observing the distribution of the data.

Keim and Kriegel [7] and Keim [8] compared the different techniques for visualizing data mining, and Kennedy et al. [9] presented a framework for information visualization. Fukuda et al. [4] proposed the SONAR system which discovers association rules from two dimensional data by visualizing the original data and finding an optimized rectangular or admissible region. Han and Cercone [6] implemented the DVIZ system for visualizing various kinds of knowledge. The basic idea of visualizing association rules in the AViz system is based on [4,6].

3 The AViz System

AViz is an interactive data mining visualization system, which uses visualization techniques to clean and preprocess the data and also interpret the patterns discovered. AViz exploits numerical attributes discretization approaches and mining algorithms to discover numerical association rules according to requirements (support threshold and confidence threshold) specified by the user. AViz consists of six components, shown in Fig. 1.

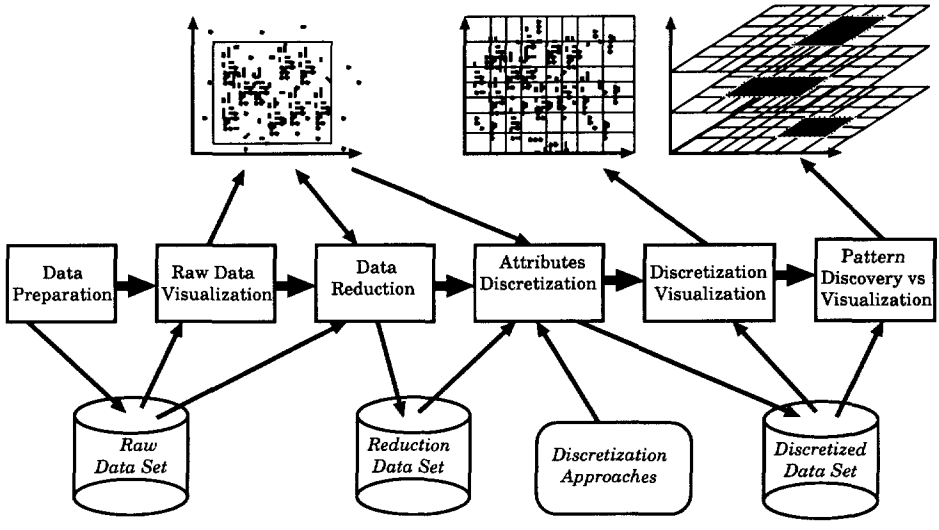


Fig. 1. The AViz System

In the AViz system, data preparation specifies the original data file, attributes file, the numerical attributes X and Y, and the nominal or numerical attribute Z, which forms the antecedence and the consequence of the association rules

to be discovered, respectively. This specification is interactively given by the user and implemented by using file dialog and choice windows. The data set prepared for discovering association rules is a list of tuples consisting of three fields $< x, y, z >$.

The second step, visualizing the raw data, reads the tuples in the data file, and transfers each tuple into a point of the drawable window. Since we only consider two numerical attributes in the antecedence of association rules, we project points in the three dimensional space $X \times Y \times Z$ onto the $X \times Y$ plane. Thus we can observe how the data distribute in the space. The denser are the points in a region, the more support is shown for the region.

According to the visualization of the raw data, the user can pick up an interesting region on the $X \times Y$ plane by using a "rubber band". This region usually contains dense points and has high support. The points outside the region are "cleaned". Therefore, the size of the data set used to discover association rules is reduced. This task is accomplished in the third step, data reduction. Another reduction is *attribute selection*, which has been completed in the data preparation step manually. The result of the reduction is redisplayed on the screen window so that reduction can take place further. This step can be repeated until the user is satisfied with the final result.

The next step is to discretize the numerical attributes, dividing each continuous attribute into disjoint intervals (buckets). AViz provides three approaches in its *Discretization Approaches* library, shown in Fig. 1, including equi-sized, bin-packing based equi-depth, and interaction-based approaches.

Discretizing numerical attributes results in two sets of intervals, one for each numerical attribute. Thus a collection of squares is obtained and stored in the *Discretized Data Set* in Fig. 1, each *square* consisting of two intervals, one from each numerical attribute. Assume that attribute X is partitioned into N_x buckets, and Y into N_y buckets, then the total number of squares is $N_x \cdot N_y$. Usually, N_x and N_y are between 20 and 300 in practice. Hence the data set is mapped into $N_x \cdot N_y$ squares, regardless of the data set size. To visualize the discretized numerical attributes, the raw data is read again from disk, and the support and hit for each square are calculated. The support of a square is the number of points which fall in it, and the hit of a square for $Z = z$ is the number of points that fall in this square and have value z of Z. For each square, the sum of its all hits corresponding to different values of Z is equal to its support. The visualization of the discretized attributes is to render all squares for all values of Z according to their support and hit, and draw and rotate a series of planes, see Fig. 4.

Finally, the algorithm for discovering the association rules is executed to find the optimal region in terms of the user-specified support and confidence threshold by moving threshold sliders, and each rule is represented as an optimal region on the plane $Z = z$ of the three-dimensional space $X \times Y \times Z$, see Fig. 5.

4 Discretizing Numerical Attributes

The AViz system provides three approaches to discretizing numerical attributes, equi-sized, bin-packing based equi-depth, and interaction-based approaches. More approaches could be added, however, in the future to compare the performance of the AViz system.

The *equi-sized approach* partitions the continuous domain into intervals with equal length. For example, if the domain of attribute *age* is $[0, 100]$, then it can be divided into small intervals with length of 10, thus we have intervals $< age, 0, 10 >, < age, 11, 20 >, \ldots, < age, 91, 100 >$. This approach is simple and easily implemented. The main drawback of this approach is it may miss many useful rules since the distribution of the data values is not considered.

Suppose the domains of numerical attributes X and Y are $[Min_X, Max_X]$ and $[Min_Y, Max_Y]$, respectively. $X \times Y$ forms an Euclidean plane. Each tuple t in the data set can be mapped to a point $(t[X], t[Y])$ in $X \times Y$. Assume X and Y are discretized into N_x and N_y buckets, respectively. Then the size of buckets is, on average, $(Max_X - Min_X)/N_x$ for X, and $(Max_Y - Min_Y)/N_y$ for Y. For a region P in $X \times Y$, we say a tuple t meets condition $(X, Y) \in P$ if t is mapped to a point in region P.

The second discretization approach used in AViz is called *bin-packing based equi-depth approach*, which is different from existing approaches. The domain of the numerical attributes may contain an infinite number of points. To deal with this problem, KID3 employs an *adjustable buckets method* [12], while the approach proposed in [13] is based on the concept of partial completeness measure. The drawback of these approaches is in time-consuming computation and/or large storage requirements. AViz exploits a simple and direct method, which is described as follows.

Assume the window size used to visualize the data set is M (width or height) in pixels, and each pixel corresponds to a *bin*. Thus we have M bins, denoted $B[i], i = 0, \ldots, M-1$. We map the raw data tuples to the bins in terms of the mapping function. Suppose $B[i]$ contains $T[i]$ tuples, and further, the attribute is to be discretized into N buckets. According to the equi-depth approach, each bucket will contain $d = \sum_{i=0}^{M-1} T[i]/N$ tuples. We first assign $B[0], B[1], \ldots$, to the first bucket until it contains at least d tuples, and then assign the following bins to the second bucket. We can repeat this operation until all buckets contain a roughly equal number of tuples.

One benefit of the bin-packing based equi-depth approach is that the storage requirement is only $O(M+N)$, depending on the number of buckets and the size of the visualization window, regardless of the domain of the attributes and the size of the data set. Another advantage of this approach is that sorting the data is not needed and the algorithm execution time is linear in the size of the data set. This method, however, may not produce enough buckets, because each bin must be assigned to only one bucket, and cannot be broken up. For instance, if the data concentrates in several bins, then the buckets that contain these bins will contain many more tuples than others. This case could happen especially when the visualization window has a small size.

The third discretization approach that AViz employs is *interaction-based.* This method consists of two steps. First, the user can specify one of the two approaches described above to simply discretize the attributes. AViz displays the discretization result. In the second step, the user can observe the data distribution and attribute discretization, and then intuitively move discretization lines to wherever he/she thinks appropriate by clicking and dragging the mouse. In this interaction process, the user can actively decide the discretization of numerical attributes. Thus, this approach can only be used with other two approaces to adjust the discretization results. However, since the visualized data has been preprocessed and mapped onto the screen, the user can only observe the graphics to obtain an approximate idea about the data distribution. For a small visualization window, distortion inevitably occurs. This may cause discretization errors.

5 Discovering and Visualizing Association Rules

In order to visualize association rules using geometric techniques, we must pursue an interesting projection of association rules to display. The basic idea is to find a small region, such as a rectangle, on the display for each association rule and use the size, color hue and intensity of each region to represent the corresponding association rules.

AViz is based on the two-dimensional model for visualizing numerical association rules proposed by Fukuda et al. [4]. The AViz system, however, extended the two-dimensional model to the three-dimensional space. Suppose the domains of numerical attributes X and Y are discretized into N_x and N_y buckets respectively. These buckets may or may not be equi-sized, depending on the discretization approach. The screen axes are partitioned correspondingly. Thus the $X \times Y$ plane is divided into $N_x \cdot N_y$ *unit squares.* A tuple t in the data set is projected to the unit square containing the point $(t[X], t[Y])$.

Consider the unit square G_{ij}, $1 \leq i \leq N_y$ and $1 \leq j \leq N_x$, which is composed of the i-th interval of Y and the j-th interval of X. Let u_{ij} denote the number of total tuples in G_{ij} and v_{ij}^z the number of tuples satisfying $Z = z$ in G_{ij}. The square G_{ij} for $Z = z$ is denoted by G_{ij}^z, and its confidence can be easily calculated as $Confidence(G_{ij}^z) = v_{ij}^z / u_{ij} \in [0, 1]$. Thus, G_{ij}^z is rendered with color RGB=$(v_{ij}^z, u_{ij} - v_{ij}^z, 0)$. The *red* component is v_{ij}^z, representing the square confidence, while the *green* component is $u_{ij} - v_{ij}^z$, and the *black* component is 0. Thus the brightness level is u_{ij}, the support of the square. So the redder the square, the higher its confidence , and the brighter the square, the higher its support.

The concepts *confidence* and *support* for a square can be extended to any form of region on the plane. The support of a region is defined as the summation of supports of all squares in the region. The confidence of a region is similarly defined.

A region is said to be *ample* if its support is greater than or equal to the support threshold. A region is said to be *confident* if its confidence is greater than or equal to the confidence threshold.

The algorithm for discovering numerical association rules by visualization is discussed in [4]. For a given confidence threshold θ, the *gain* of the square G_{ij}^z is defined as

$$gain(G_{ij}^z) = v_{ij}^z - \theta \times u_{ij}.$$

Obviously, when the confidence of $G_{ij}^z = \theta, gain(G_{ij}^z) = 0$. The gain reflects the confidence. When the confidence is greater than the threshold, the gain is positive, while the gain is negative when the confidence is less than the threshold. The *gain* of a region is the gain summation of all squares in the region.

We implement a dynamic programming algorithm for finding the optimized gain rectangles in three-dimensional space, which is based on the algorithm for two-dimensional space [4]. The time complexity of our algorithm is $O(N_z \cdot N_x \cdot N_y \cdot min\{N_x, N_y\})$, where N_z is the number of values of attribute Z, if Z is nominal, or the number of discrete intervals, if Z is numerical. The basic idea is, for each Z value z, to choose randomly a pair of rows, say i-th and j-th rows, $1 \leq i \leq j \leq N_y$, and consider rectangles $G^z([i,j],m)$ on the plane $Z = z$, which consists of the squares from the i-th row to the j-th row in the m-th column, for $m = 1, 2, \ldots, N_x$. Then compute the gain for each rectangle $G^z([i,j],m)$,

$$gain(G^z([i,j],m)) = \sum_{k=i}^{j} gain(G_{km}^z) = \sum_{k=i}^{j} (v_{km}^z - \theta \times u_{km}).$$

Finally, compute the gain for rectangular regions $G^z([i,j],[r,k])$, $1 \leq i \leq j \leq N_y, 1 \leq r \leq k \leq N_x$, which consists of rows from i to j, and columns from r to k on the plane $X \times Y$ with $Z = z$. The optimized gain rectangle in the plane $Z = z$ is one with the highest gain.

6 AViz Implementation and Experiment

The AViz system has been implemented in JDK1.2 and Java3D. The data preparation is accomplished by choosing a data file and attributes file, and specifying the attributes to be mined. The data file is formatted in tuples which consists of a series of fixed length fields. The attributes file characterizes each attribute, including attribute name, type, length, position in the tuple, and domain. This is implemented in dialog windows (under the file menu and setting menu). The steps of discovering knowledge are controlled by the *control menu*, which consists of the next five steps. Two sliders are used to control the support threshold and confidence threshold. By moving these sliders in the discovering step, the resulting rules (focus area in each planes parallel with $X \times Y$ plane) can vary.

AViz has been applied to the U.S. census data in 1997 to find the association rules between attributes. The data set contains about 1.4 million tuples, each tuple consisting of 5 numerical attributes *age, total-person-income, taxable-income-amount, tax-amount, hours-usually-worked-per-week* and 3 nominal attributes *sex, race, class-of-work.*

In the following we give an example to trace the process of association rules discovery and visualization.

Step 1: Data preparation

Choose two numerical attributes X=*Taxable-income-amount* and Y=*Total-person-income*, and an nominal attribute Z=*Race*. The domain of X and Y is $[0, 1000K]$ and $[0, 500K]$, respectively. Z takes the following five values: *White, Black, Amer-Indian-or-Eskimo, Asian-or-Pacific-Islander*, and *Other*. Also, we specify that X and Y are to be discretized into 20 intervals.

Step 2: Raw data visualization

Project the raw data onto the visualization window, shown in Fig. 2. In the mapping, the Z value of the tuples is not considered, and only X and Y values are regarded.

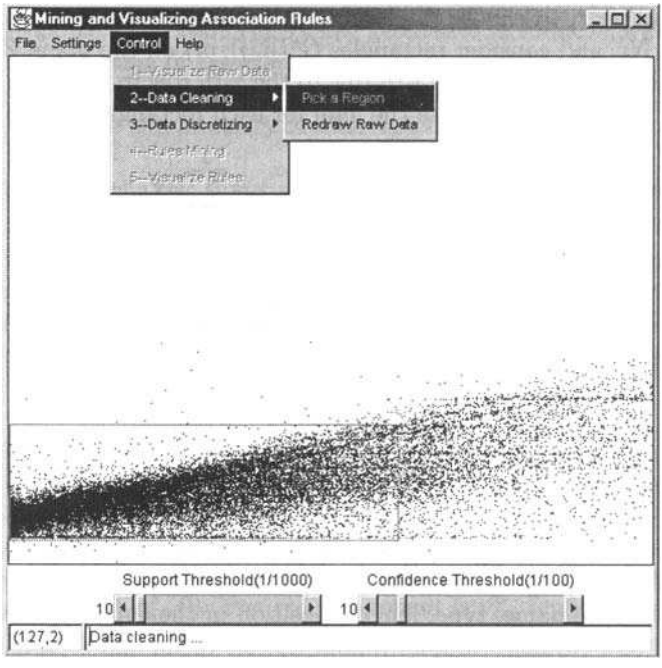


Fig. 2. The raw census data and interesting area

Step 3: Data cleaning

From Fig. 2, we see that most data concentrates on a strip which is interesting to us. The other data can be cleaned. For now, we pick this strip by using a rubber band. After cleaning, the remaining data set contains about 1.08 million tuples.

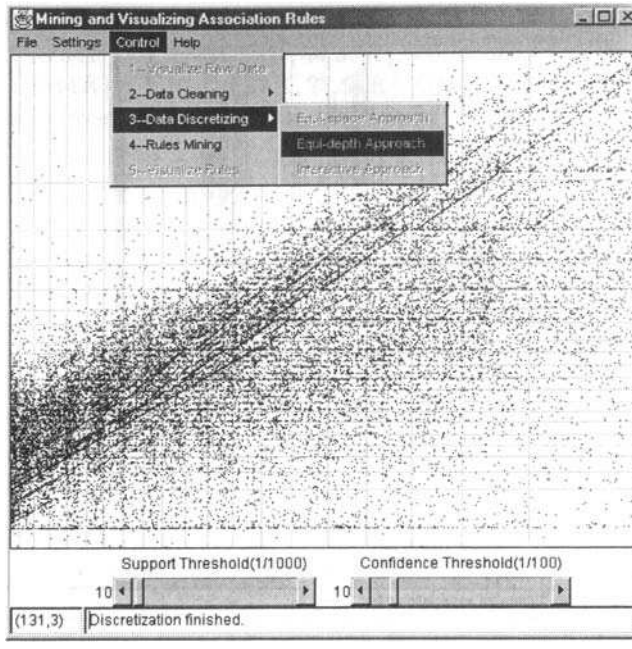


Fig. 3. Discretizing the interesting area of numerical attributes

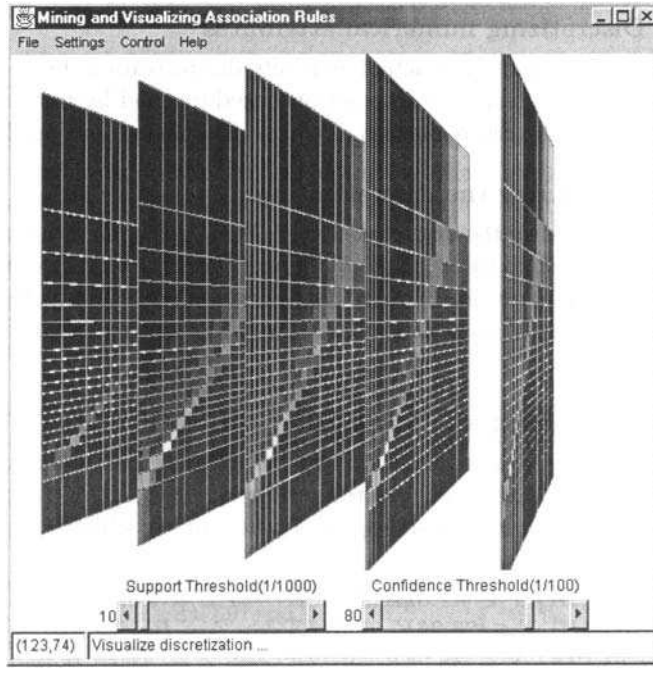


Fig. 4. Visualizing the discretization by rotating planes around Y axis

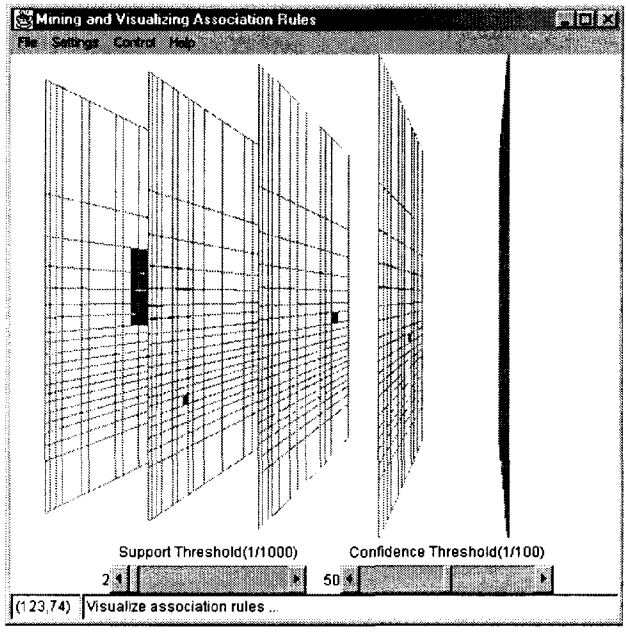


Fig. 5. Discovering the Optimal Rectangles representing association rules

Step 4: Discretizing numerical attributes

We choose the second approach of attribute discretization, bin-packing based equi-depth, and then utilize the interaction-based method by moving discretization lines to adjust the discretization. The result is shown in Fig. 3

Step 5: Visualizing the discretization

In Fig. 4 we visualize the discretization of *Taxable-income-amount* and *Total-person-income* for each value of Z = *White, Black, Amer-Indian-or-Eskimo, Asian-or-Pacific-Islander*, and *Other*. Each Z value corresponds to a plane and the volume consisting of all planes rotates around Y axis so that all planes can be viewed clearly.

Step 6: Discovering and Visualizing association rules

To find the association rules, we move the threshold sliders and specify the support threshold and confidence threshold as 0.2% and 50%, respectively. We obtain five rules, each corresponding to a value of Z, which are described as follows and shown in Fig. 5.

$X \in [15.64K, 18.34K], Y \in [41.62K, 44.58K] \Longrightarrow Z =$ *White*
$X \in [9.78K, 10.44K], Y \in [26.26K, 28.51K] \Longrightarrow Z =$ *Black*
$X \in [12.67K, 13.12K], Y \in [38.24K, 39.72K] \Longrightarrow Z =$ *Amer-Indian-or- Eskimo*
$X \in [11.99K, 12.72K], Y \in [33.12K, 33.73K] \Longrightarrow Z =$ *Asian-or-Pacifi c-Islander*
$X \in [10.66K, 11.12K], Y \in [30.20K, 31.13K] \Longrightarrow Z =$ *Other*

The result shows that *White* has the largest optimal and most upper area $X \in [15.64K, 18.34K] \wedge Y \in [41.62K, 44.58K]$, while *Black* has the lowerest area $X \in [9.78K, 10.44K] \wedge Y \in [26.26K, 28.51K]$. Other three categories have the optimal areas between those of *White* and *Black*.

7 Concluding Remarks

AViz is an interactive system for visualizing and discovering numerical association rules. The basic idea is to use visualization techniques to constrain the domain of data by interacting with user and then to discover rules from the reduced data, and finally to visualize the resulting knowledge. In our implementation, we emphasize the human-machine interaction, since we believe that interactive visualization plays an important role in data mining to guide the process of discovering knowledge. The experiment by visualizing a large data set has also demonstrated that it is useful for users to understand the relationships among data and to concentrate on the meaningful data to discover knowledge. The capability of the AViz system will be expanded to visualize not only the process of discovering association rules but also the dynamic processes of discovering other kinds of knowledge, like classification rules. Another problem is about high-dimensional data. An approach for reducing the dimensionality of the original data based on the *principal coordinate analysis* is being considered. Combining the visualization and data mining algorithms will produce a much more efficient method of knowledge discovery.

Acknowledgments

The authors are members of the Institute for Robotics and Intelligent Systems (IRIS) and wish to acknowledge the support of the Networks of Centers of Excellence Program of the Government of Canada, the Natural Sciences and Engineering Research Council, and the participation of PRECARN Associates Inc.

References

1. R. Agrawal, R. Srikant, Fast algorithm for mining association rules in large databases, Proc. of the 20th International Conference on VLDB, pp.487-499, Sept. 1994.
2. Y. Cai, N. Cercone, J. Han, Attribute Oriented Induction in Relational Databases, in *Knowledge Discovery in Databases* ed. by Gregory Piatetsky-Shapiro and William J. Frawley, pp.213-228, 1991.
3. M. Derthick, J. Kolojejchick, S. F. Roth, An Interactive Visualization Environment for Data Exploration, KDD-97, pp.2-9, 1997.
4. T. Fukuda, Y. Morimoto, S. Morishita, T. Tokuyama, Data Mining Using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization, Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 13-24, 1996.

5. R. Groth, Data Mining: A Hands-on Approach for Business Professionals, Prentice Hall PTR, 1998.
6. J. Han, N. Cercone, DVIZ: A System for Visualizing Data Mining, Proc. of the 3rd Pacific-Asia Knowledge Discovery in Databases, pp. 390-399, 1999.
7. D. A. Keim, H. P. Kriegel, Visualization Techniques for Mining Large Databases: A Comparison, Transactions on Knowledge and Data Engineering, Dec.,1996.
8. D. Keim, Vizual Data Mining, Proceedings of International Conference on Very Large Data Bases, 1997.
9. J. B. Kennedy, K. J. Mitchell, P. J. Barchay, A framework for information visualization, SIGMORD Record, Vol.25, No.4, Dec., 1996.
10. B. Liu, W. Hsu, K. Wang, S. Chen, Visually Aided Exploration of Interesting Association Rules, Proc. of the 3rd Pacific-Asia Knowledge Discovery in Databases, pp. 380-389, 1999.
11. R. J. Miller, Y. Yang, Association Rules over Interval Data, Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 452-461, 1997.
12. G. Piatetsky-Shapiro, Discovery, Analysis, and Presentation of Strong Rules, in *Knowledge Discovery in Databases* ed. by Gregory Piatetsky-Shapiro and William J. Frawley, pp.229-260, 1991.
13. R. Srikant, R. Agrawal, Mining Quantitative Association Rules in Large Relational Tables, Proc. of the ACM SIGMOD International Conference on Management of Data, pp.1-12, 1996.

Discovering Unordered and Ordered Phrase Association Patterns for Text Mining

Ryoichi Fujino[1], Hiroki Arimura[2,3], and Setsuo Arikawa[3]

[1] Nippon Steel Information and Communication Systems, Inc.
Kitakyushu 804-0001, Japan
[2] PRESTO, Japan Science and Technology Corporation
[3] Dept. Informatics, Kyushu Univ., Fukuoka 812-8581, Japan,
{arim,fujino,arikawa}@i.kyushu-u.ac.jp

Abstract. This paper considers the problem of finding all frequent phrase association patterns in a large collection of unstructured texts, where a phrase association pattern is a set of consecutive sequences of arbitrary number of keywords which appear together in a document. For the ordered and the unordered versions of phrase association patterns, we present efficient algorithms, called Levelwise-Scan, based on the sequential counting technique of Apriori algorithm. To cope with the problem of the huge feature space of phrase association patterns, the algorithm uses the generalized suffix tree and the pattern matching automaton. By theoretical and empirical analyses, we show that the algorithms runs quickly on most random texts for a wide range of parameter values and scales up for large disk-resident text databases.

1 Introduction

Background. Recent progress of network and storage technologies have been rapidly increasing the size and the species of *text databases* such as webpages, SGML/XML archives, and a collection of emails or text files accumulated on a file system. These lead to potential demands for new access methods for large text databases. However, traditional data mining research mainly considers well-structured databases like transaction databases with boolean or numeric attributes [2,3,9,16], and thus there still are a small number of studies on text data mining [5,6,10,14,18].

Phrase association patterns. We consider the problem of discovering all frequent patterns in a large collection of unstructured texts. Suppose we are given a set of documents S. A *phrase* over S is any string of arbitrary length that appears in S. We do not assume any semantic restriction on phrases as used in natural language processing, and thus we mean by a phrase merely a sequence of keywords delimited by space letters. An *unordered k-proximity phrase association pattern* (unordered phrase pattern, for short) is an unordered set of phrases associated with a nonnegative integer k, called a proximity, such as $(\{\langle\text{data mining}\rangle, \langle\text{very large databases}\rangle\}, 30)$ which expresses a pattern that

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 281–293, 2000.
© Springer-Verlag Berlin Heidelberg 2000

the phrases ⟨data mining⟩ and ⟨very large databases⟩ appear in a document close to each other, more precisely within the distance k. Similarly, an *ordered k-proximity phrase association pattern* (ordered phrase pattern, for short) is defined except that phrases must appear in a specified order. For example, ((⟨a silkworm missile⟩, ⟨attacks⟩, ⟨iranian oil platform⟩), 30) is an ordered k-proximity phrase association pattern. In information retrieval [7], queries consisting of phrases are proved to be more powerful than conventional queries consisting of keywords.

The problem we consider is the *frequent pattern problem*, which is the problem to find all phrase association patterns that appear more than a user-specified threshold. An efficient method for solving this problem can be used for finding interesting patterns based on various information-theoretic measures such as the confidence [3], the information entropy [16], and the classification accuracy[6].

Approaches. If the maximum number of phrases in a pattern is bounded by a constant d then the frequent pattern problem for both unordered and ordered patterns is solvable by *Enumerate-Scan* algorithm [18], a modification of a naive generate-and-test algorithm, in $O(n^{d+1})$ time and $O(n^d)$ scans although it is still too slow to apply real world problems. Arimura *et al.* [6] gives the *Split-Merge* algorithm that finds frequent ordered phrase patterns in almost linear time with poly-log factor for random text databases. However, this algorithm is inefficient for large disk-resident text databases, and hard to extend for unordered patterns. As related researches, [10] considered the discovery of association rules over keywords with predefined category-tags, and [14] studied data mining of *episodes*, a non-consecutive sequence of events. Unfortunately, these methods are not directly applicable to our problem.

A possible approach is to follow the design principle of the *Apriori algorithm* [3], the state-of-the-art algorithm for mining association rules in transaction databases. However, a problem in this approach is that the feature space of phrase patterns is huge compared with that for transaction databases. For example, even a subset of *Reuters newswires* [17] of size 460KB contains two hundred thousands of unique phrases (Section 5) while a typical database contains less than two thousands of attributes in the basket analysis ([3]). Hence, we require efficient handling of the huge feature space of phrase patterns.

Main results. In this paper, we present a practical algorithm, called *Levelwise-Scan*, for efficiently finding frequent unordered phrase patterns as well as ordered phrase patterns from a large disk-resident database based on the approach of the Apriori algorithm [3] and the Enumerate-Scan algorithm [18]. To overcome the problem of the huge feature space of phrase patterns, the algorithm incorporates the techniques of the *generalized suffix tree* [18] and the *pattern matching automaton* [4] for efficiently storing and detecting frequent phrases in a text database.

In theoretical analyses, we show that for random text databases, Levelwise-Scan quickly finds all frequent unordered d-phrase patterns in almost linear time $O(n^2 + (\log n)^d N)$ and space $O(n \log n + R)$ with a constant factor depending on

k and d, where n and N are the total sizes of a small sample and the whole text database, respectively, and R is the output size. For ordered d-phrase patterns, we have similar time complexity. The experiments on Reuters newswire test data [17] show that the Levelwise-Scan algorithm performs well for various values of parameters and scales up for large text data with linear time complexity. By simulation results, we estimate that the algorithm will run in 40 minutes for 100MB of disk-resident text data. From these results, we conclude that the proposed algorithm is efficient in practice as well as in theory for large text databases.

2 Formulation

Text databases. Let Σ be a finite alphabet of symbols. In this paper, we assume the alphabet $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \ldots, \mathrm{z}, \ldots, *, +, ",", ".", "\sqcup"\}$, including the space symbol "$\sqcup$". We denote by Σ^* the set of all finite strings over Σ, and by ε the empty string. For a string s of length n and $1 \leq i \leq n$, we denote by $|s|$ the *length* of s and by $s[i]$ the ith letter of s. For a set of strings $S \subseteq \Sigma^*$, we denote by $|S|$ the *cardinality* of S and by $||S|| = \sum_{s \in S} |s|$ the *total length* of S.

If there exist strings $u, v, w \in \Sigma^*$ for a string $t \in \Sigma^*$ such that $t = uvw$ then u, v and w are a *prefix*, a *substring*, and a *suffix* of t, respectively. For a string $s, t \in \Sigma^*$, if there exists a positive integer $1 \leq i \leq |t|$ such that $t[i] \cdots t[i+|t|-1] = s$ then s *occurs in* t *at* i. The integer i is called an *occurrence of* s *in* t.

A *document* is any string $s \in \Sigma^*$, and a *text database* is a finite collection $T = \{s_1, \ldots, s_m\}$, $m \geq 0$, of documents. In our definition, a *phrase* is any substring π of $d \in T$ which may delimited with spaces "$\sqcup$." For example, ⟨Lloyd's␣Shipping␣Intelligence⟩ is a phrase.

Phrase association patterns. Let d and k be any nonnegative integers. An *unordered k-proximity d-phrase association pattern* (or *unordered (k,d)-phrase pattern*) is a pair $\pi = (\{p_1, \ldots, p_d\}, k)$ of an unordered set $\{p_1, \ldots, p_d\} \subseteq \Sigma^*$ of d phrases and a parameter k called a *proximity*. The pattern π *appears in* a document s if there exist some substrings $s_0, \ldots, s_d$ of s such that (i) $s = s_0 p_{i(1)} s_1 \cdots, s_{d-1} p_{i(d)} s_d$ for some permutation $\{i(1), \ldots, i(d)\}$ of $\{1, \ldots, n\}$ and (ii) $|s_i| \leq k$ for every $i = 1, \ldots, d-1$. Note that in our definition, the proximity k is counted in the number of letters. Similarly, an *ordered k-proximity d-phrase association pattern* (or *ordered (k,d)-phrase pattern*) [5,6] is a pair $\pi = ((p_1, \ldots, p_d), k)$ of a sequence $p_1, \ldots, p_d \in \Sigma^*$ of d phrases and a proximity k. The appearance of an ordered pattern is defined similarly except that the permutation is fixed to the identity such that $i(j) = j$ for every $1 \leq j \leq d$. We sometimes omit the proximity k if it is clear from context.

Data Mining problems. Let $T = \{t_1, \ldots, t_m\}$, $m \geq 0$, be a text database and $\mathcal{C}$ be a class of patterns (also called *hypothesis space*), where each $\pi \in \mathcal{C}$ is identified with a mapping $\pi : T \to \{0, 1\}$. The *document count* of π, denoted by $hit_T(\pi)$, is the number of documents $s \in T$ in which π appears. Then the *frequency* of π w.r.t. T is defined by $supp_T(\pi) = hit_T(\pi)/|T|$. For $0 \leq \sigma \leq 1$,

a pattern π is *σ-frequent w.r.t. T* or simply *frequent* if $supp_T(\pi) \geq \sigma$. Then, the problem we consider is a variant of the frequent pattern problem, which is originally introduced by Agrawal and Srikant [3] and extensively studied for transaction databases.

Frequent Pattern Problem for class $\mathcal{C}$
Input: A text database $T \subseteq \Sigma^*$, and a threshold $0 \leq \sigma \leq 1$ for frequency.
Problem: Find all frequent patterns $\pi \in \mathcal{C}$ w.r.t. T, i.e., $supp_T(\pi) \geq \sigma$.

Sampling is a useful technique to cope with large disk-resident text databases. A *sample* is any subset $S \subseteq T$. An *S-induced pattern* is a pattern π such that every phrase of π appears at least once in S. To capture the data mining with sampling, we introduce a modified version of the problem stated as follows.

Frequent Pattern Problem with sampling for class $\mathcal{C}$
Input: A text database T, a sample $S \subseteq T$, and a threshold $0 \leq \sigma \leq 1$.
Problem: Find all S-induced patterns $\pi \in \mathcal{C}$ that is frequent w.r.t. T, i.e., $supp_T(\pi) \geq \sigma$.

3 Previous Algorithms

In this section, we briefly review two existing algorithms Enumerate-Scan [18] and Split-Merge [5,6] for solving the frequent pattern problem. Then, in the next section, we will present a new algorithm Levelwise-Scan. Algorithms other than Split-Merge can deal with both the unordered and the ordered versions. In what follows, n denotes the total size of the input text database T and d be the number of phrases in a pattern.

3.1 Enumerate-Scan Algorithm

The *Enumerate-Scan* algorithm implements a naive *generate-and-test* method [18]. Enumerating all of $O(n^{2d})$ possible patterns, the algorithm counts the frequency of each pattern by scanning the entire text database. If the number of phrases is bounded by constant $d \geq 0$, the algorithm solves the frequent pattern problem in $O(n^{2d+1})$ time with $O(n^{2d})$ disk scans for unordered and ordered cases. Wang *et al.* [18] showed that the complexity can be improved to $O(n^{d+1})$ time and $O(n^d)$ scans by using the generalized suffix tree (See Sec. 4.2) although it still requires several hours for mining genome sequences of 15KB for 2-phrase patterns with approximate matching [18].

3.2 Split-Merge Algorithm

In the previous works [5,6], we developed an efficient algorithm, here called *Split-Merge*, that finds all the optimal patterns for the class of ordered (k, d)-proximity phrase patterns. The Split-Merge algorithm quickly runs in almost

Algorithm Levelwise-Scan;
Input: A text database $T = \{s_1, \ldots, s_m\}$, a frequency threshold $0 \leq \sigma \leq 1$.
Output: The set L_i of σ-frequent i-phrase patterns w.r.t. T for every $0 \leq i \leq d$.

Phase A (*Building Phase*) In the first pass, the algorithm do the followings.
- A.1 First, draw a small sample set S from T that fits in main memory if sampling is required.
- A.2 Build the generalized suffix tree (GST) for S.
- A.3 Count the occurrence of each phrase to select frequent phrases using the GST (explained in Sec. 4.2).
- A.4 Build a finite automaton M for recognizing L_1.

Phase B (*Scanning Phase*) In the following pass $i = 2, \ldots, d$, the algorithm computes the set L_i of all frequent i-phrase patterns by iterating the following process until no more frequent patterns are found.
- B.1 First, it builds the candidate set C_i of i-phrase patterns using the frequent patterns in L_{i-1} built in pass $i - 1$.
- B.2 Next, the algorithm scans the whole text database T from left to right using M to count the frequencies of all patterns in C_i at a single pass on T and to build L_i, the set of all frequent i-phrase patterns.

Fig. 1. Levelwise-Scan algorithm

linear time $O(k^{d-1} n \log^{d+1} n)$ with poly-log factor using $O(k^{d-1} n)$ space for most nearly random texts. The algorithm quickly searches the hypothesis space using dynamic reconstruction of the content index, called a *suffix array*. Kasai *et al.* [11] reported that the algorithm finds the best 600 patterns at the information entropy in a few minutes for a subset of Reuters newswires [17] of 7 mega-bytes using a few hundreds mega-bytes of main memory.

4 Levelwise-Scan Algorithm

4.1 Outline of the Algorithm

In Fig. 1, we present our algorithm *Levelwise-Scan*, which efficiently finds all frequent unordered and ordered versions of phrase patterns in a given collection of documents. The Levelwise-Scan algorithm is based on the same design principle as the *Apriori algorithm* of Agrawal *et al.* or *Levelwise algorithm* in [2,3]. Based on a similar strategy employed by the Apriori algorithm [3], the Levelwise-Scan algorithm runs through several passes over a text database T to compute the set L_i of all frequent i-phrase patterns for every $i = 1, \ldots, d$.

In the following sections, we will describe each phase in more detail. For simplicity, we consider the case without sampling, i.e., $S = T$. In Section 4.4, we will discuss the efficiency with sampling. In what follows, $S = T = \{s_1, \ldots, s_m\}$ is a text database of $m \geq 0$ texts and $n = ||S||$ is the total size of S. We denote by $0 \leq \sigma \leq 1$ the minimum frequency threshold and by d and k the number of phrases and the proximity of a pattern to be found, respectively. Let denote by l the length of the longest frequent phrases in L_1.

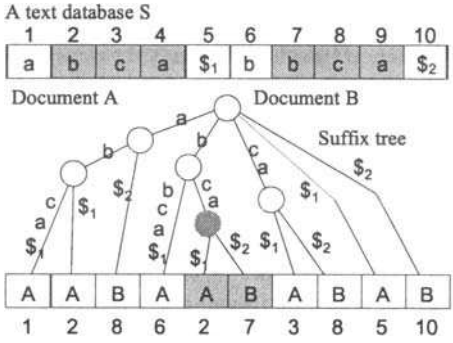


Fig. 2. The generalized suffix tree (GST) ST for $S = \{abca\$_1, bbca\$_2\}$. The shadowed node corresponds to the substring bca and two shadowed leaves correspond to the two occurrences of bca. Suffix links are omitted. (Sec. 4.2)

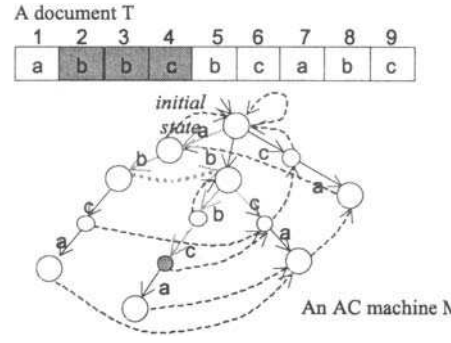


Fig. 3. The AC-pattern matching machine M for $P = \{abca, bbca, bca, ca, a\}$, which corresponds to all internal nodes of the GST in Fig. 2. The solid and the dotted lines indicate goto and failure functions, respectively. (Sec. 4.3)

4.2 Generalized Suffix Trees

We use the *generalized suffix tree* (*GST*, for short) for efficiently storing all phrases appearing in S (Step A.2 of Fig. 1). First, we assume that every document $s_i \in S$, $1 \leq i \leq m$, is terminated with appending a special delimiter symbol $\$_i$ such that $\$_i \neq c$ for any $c \in \Sigma \cup \{\$_j\}$ $(i \neq j)$. The GST for S is the *compacted trie* for all suffices of $s \in S$, which is obtained from the (uncompacted) trie for all suffices [1] by removing all internal nodes with a single child and concatenating its labels into a single string [15,18]. Each node v of a GST represents the string $word(v)$, called a *branching substring* w.r.t. S, which is obtained by concatenating the labels on the path from the root to v in its order. Each leaf is labeled with the name of the document it belongs to. Each internal node has the *suffix link* which points to the node w such that $word(v) = c \cdot word(w)$ for some $c \in \Sigma$. In Fig. 2, we show an example of the GST.

The GST ST for S uses $17n$ bytes to store and can be built in $O(n)$ time [15]. It is known that the maximum length l of the branching substrings and the height h, denoted by $height(ST)$, are both $O(\log n)$ for random texts [8]. We define the set $Occ_S(\pi) = \{\, s \in S \mid \pi \text{ occurs in } s \,\}$. Then, patterns π and τ to be *equivalent* if $Occ_S(\pi) = Occ_S(\tau)$.

Definition 1. *An unordered or ordered phrase pattern π is* of canonical form *w.r.t. $S \subseteq \Sigma^*$ if it consists only of branching substrings w.r.t. S.*

Lemma 1 (Arimura *et al.* [5]). *For any unordered (or ordered k-proximity) d-phrase pattern π that appears at least one document in S, there exists a pattern in the same class that is of canonical form and equivalent to π.*

Procedure Count-Prune;
Input: The GST ST for sample S and the frequency threshold $0 \leq \sigma \leq 1$;
1. For each node v, do (a)–(c) below: /* in the depth-first order */
   (a) If v is a leaf with label i, then set all but the i-th bit to be zero.
   (b) If v is an internal node and we returned from a child w then $bits(v) = bits(v) + bits(w)$, and discard $bits(w)$.
   (c) If $\#bits(v) \geq \sigma \cdot |S|$, then mark v as frequent.
2. Prune all unmarked nodes by traversing ST.

Fig. 4. The procedure for finding all frequent phrases and prune the GST

4.3 Building Phase

This phase is common to both unordered and ordered versions of the algorithm. In the building phase (Phase A) of Fig. 1, the algorithm finds all σ frequent phrases and stores them in a space efficient data structure called the generalized suffix tree.

Computing frequent phrases. In Fig. 4, we show the procedure Count-Prune for computing the set L_1 and then pruning the GST, which implements Step A.3 of Fig. 1. Count-Prune first solve the problem of counting the frequency of all branching substrings using the GST ST for S, which is known as the *color set size problem* [12].

Although this problem has an $O(n)$ time solution [12], we use a simpler algorithm for its practical efficiency. In Fig. 4, each node v has a bit-vector $bits(v)$ of length m such that the i-th bit is on iff $word(v)$ occurs the i-th document. We denote by $\#$ and $+$ the bitcount and the bitwise-or operators, respectively.

Lemma 2. *The procedure Count-Prune in Fig. 4, given the GST ST for S, computes the set L_1 of all frequent phrases w.r.t. S in time $O(mn)$ and $O(mh)$ additional space, where $h = height(ST)$.*

The following lemmas justify our pruning strategy similar to [3]. Let $0 \leq \sigma \leq 1$.

Lemma 3. *Let v, w be nodes of GST for $S \subseteq \Sigma^*$. Then, if v is an ancestor of w then $hit_S(word(v)) \geq hit_S(word(w))$, where $hit_S(p) = \{ s \in S \mid p \text{ appears in } s \}$.*

Lemma 4. *If an unordered (or ordered) d-phrase pattern π is σ-frequent w.r.t. S then so is the $(d-1)$-phrase pattern obtained from π by removing any phrase.*

Building a pattern matching machine over GST. At Step A.4 of Fig. 1, Levelwise-Scan builds a finite state automaton M, called an *Aho-Corasick pattern matching machine* (*AC-machine*, for short) [4] for recognizing L_1.

Let $P \subseteq \Sigma^*$ be a set of strings. Then, the AC-machine M for P is exactly the (uncompacted) trie $UT(P)$ [1] for the strings in P augmented with the failure and the output functions attached to each node. The *initial state* of M is the root. The *goto function* is the set of labeled edges forming the trie $UT(P)$. As in the suffix tree, each node v represents the string $word(v)$. The *failure function* of

v is defined to be the edge from v to the unique node that represents the longest suffix of $word(v)$ which is a prefix of some string in P. The *output function* of v is the set of all the suffices of $word(v)$ contained in P. In Fig. 3, we show an example of the AC-machine M. After scanning the first four letters $abbc$ of a given text $T = abbcbcabc$, M detects the strings bbc, bc, c by following the path $(a, b, fail, b, c)$ from the root to the shaded node.

Lemma 5 (Aho and Corasick [4]). *We can construct the AC-machine for P in $O(||P||)$ time, and detect the occurrence of all strings $p \in P$ in a given document of length n in $O(n)$ time by scanning the document once.*

A set $P \subseteq \Sigma^*$ is *substring-closed* if $P = \{ s \in \Sigma^* \mid t \in P, s \text{ is a substring of } t \}$. By the next lemma, we can construct the AC-machine for L_1 directly on the GST for L_1 at Step A.4 of Fig. 1.

Lemma 6. *Let $P \subseteq \Sigma^*$ be substring-closed. Then, the edges and the suffix links of the GST for P, respectively, are isomorphic to the goto and the failure functions of the AC-machine for P at all branching nodes. (See* Fig. 2 *and* Fig. 3.*)*

4.4 Scanning Phase

In the scanning phase (Phase B) of Fig. 1, Levelwise-Scan counts the occurrences of all frequent patterns of L_i by scanning the text T from left to right.

Candidate generation. In Step B.1, we build the *candidate set* C_i from L_{i-1}. This part is almost same as that of [3]. For each $i > 1$, Levelwise-Scan constructs the i-th candidate set C_i by merging members of L_{i-1} and pruning many non-frequent members of C_i using L_{i-1} on memory using Lemma 3 and Lemma 4. To store each branching phrase, encoded with the pointer to a node of the GST, we use a hash-based trie (called the *hash tree* in [3]).

Counting: the unordered version. In Fig. 5, we show the procedure Unordered_Scan for the unordered case, which implements Step B.2 of the Levelwise-Scan algorithm. Let $t \in T$. Scanning t from left to right, the algorithm detects the occurrence of the longest branching substring $word(v_i)$ terminating at the position $i = 1, \ldots, |t|$, where v_i is a node or a state of M. $P = (v_1, \ldots, v_n)$ is the resulting array of nodes representing such occurrences. To avoid enumerating redundant patterns, it detect only the longest matching v_i using an AC-machine M. A $(2d-1)$-tuple $(o_1, \ldots, o_d; l_1, \ldots, l_d) \in \{1, \ldots, |t|\}^{2d}$ is (k, d)-*admissible w.r.t.* P if (i) $0 \leq (o_i - l_i) - o_{i-1} \leq k$ for every $i = 2, \ldots, d$, and (ii) $0 \leq l \leq len(o_{i+1})$ for every $i = 1, \ldots, d$, where $len(o_i)$ is the length of the branching substring $word(P(o_i))$ represented by o_i. It is easy to see that if a (k, d)-phrase pattern $(\{p_1, \ldots, p_d\}, k)$ appears in t then the set of the right ends o_i of the occurrences of $p_1, \ldots, p_d$ together with a set of the lengths l_i of the phrases p_i form an admissible tuple $(o_1, \ldots, o_d; l_1, \ldots, l_d)$, and the converse also holds. There are at most $O(k^d l^d |t|)$ such k-admissible d-tuples with a given P, where l is the maximum length of branching substrings. $\mathcal{S}(p_i)$ is the set of all nodes representing the suffices of a phrase p_i, which is obtained by following

Procedure Unordered_Scan;
Input: the level $d \geq 2$, the AC-machine M for S and a text database $T = \{t_1, \ldots, t_m\}$;
Output: the set L_d of all S-induced frequent unordered (k,d)-phrase patterns w.r.t. T.
1 Set $C(\pi) = 0$ and $D(\pi) = 0$ for every $\pi \in C_d$.
2 **foreach** $t_\delta \in T$ $(\delta = 1, \ldots, m)$ **do**
3 Initialize the array P of nodes.
4 **foreach** $i = 1, \ldots, |t|$ **do**
5 If M moves the next state v reading the letter $t[i]$, then set $P(i) := v$.
6 Let $O \subseteq \{1, \ldots, |t|\}^{2d}$ be the set of all (k,d)-admissible tuples w.r.t. P.
7 **foreach** $(o_1, \ldots, o_d; l_1, \ldots, l_d) \in O$ **do**
8 Let $p_1 \in \mathcal{S}(o_1), \ldots, p_d \in \mathcal{S}(o_d)$ be the phrases satisfying $|p_1| = l_1, \ldots, |p_d| = l_d$.
9 Sort $(p_1, \ldots, p_d)$ in a total order over nodes by a permutation $j(\cdot)$.
10 **if** $\pi = ((p_{j(1)}, \ldots, p_{j(d)}), k) \in C_d$ and $\delta > D(\pi)$ **then**
11 $C(\pi) = C(\pi) + 1$ and $D(\pi) = \delta$.
12 Insert all patterns $\pi \in C_d$ such that $C(\pi) \geq \sigma \cdot |T|$ into L_d.

Fig. 5. The unordered version of the scanning phase of the Levelwise-Scan algorithm for unordered (k,d)-phrase patterns, which implements Step B.2 of the algorithm.

failure links. Furthermore, we use $C(\pi)$ to keep the document count of π, and $D(\pi)$ to keep the name of the last document for avoiding duplicated counting.

Counting: the ordered version. For the ordered case, we use the procedure Ordered_Scan, which is a modification of Unordered_Scan in Fig. 5. It is the only difference between two procedures that Ordered_Scan does not sort obtained sequence $(p_1, \ldots, p_d)$ at Line 9 before check the membership $\pi \in C$, while Unordered_Scan sorts it to have a lexicographcally smallest permutation $(p_{j(1)}, \ldots, p_{j(d)})$ as a representative.

4.5 Time Analysis

Now, we give the correctness and the time complexity of the Levelwise-Scan algorithm. Strings in an infinite family $s_1, s_2 \cdots \in \Sigma^*$ are said to be *almost random* if for any s_i, the maximum length of the branching substrings of s_i is $O(\log n)$, where $n = |s_i|$. Any random string generated by the uniform distribution over Σ is known to be almost random with high probability [8]. Genetic sequences are also known to behave like almost random strings.

With sampling. Let $T \subseteq \Sigma^*$ be a text database of total size N and $S \subseteq T$ be a sample of total size n. We denote by R the total size of the output $\sum_i ||L_i||$.

Theorem 1. *Let $k, d \geq 0$, T be a set of almost random texts and S be its sample. For the class of unordered (k,d)-phrase patterns, Levelwise-Scan of* Fig. 1 *solves the frequent pattern problem with sampling in time $O(n^2 + dk^{d-1}(\log n)^d N)$ and space $O(n \log n + R)$.*

Proof. We can show that the algorithm runs in $O(n^2 + ||L_1|| + dk^d l^d N)$ time and $O(nh + R)$ space when l is the maximum length of the branching substrings

Table 1. Performance overview: the number $\#L_2$ of patterns found and the running time *Total* for a sample of fixed size $n = 466KB$ (a) with varying *min-freq* from 0.0025 to 0.04 and (b) with varying proximity k from 10 to 100, where $k = 10$, *min-freq* $= 0.01$, and *max-freq* $= 0.5$ if not specified.

	Unordered		Ordered	
min-freq	$\|L_1\|$	Time (s)	$\|L_1\|$	Time (s)
0.0400	42	35.5	3	36.3
0.0200	136	41.7	18	41.3
0.0100	345	62.2	57	63.8
0.0050	774	130.7	118	134.4
0.0025	1,419	314.1	184	324.5

	Ordered	
proximity	$\|L_1\|$	Time (s)
10	4	4.0
20	32	32.0
40	84	84.0
80	176	176.0
100	204	204.0

in T and h is the height of the GST for S. If T and thus S are almost random then l and h are both $O(\log n)$. R is the order of $dk^d l^d N$. □

Corollary 1. *Let $k, d \geq 0$. For the class of ordered (k, d)-phrase patterns, the frequent pattern problem with sampling is solvable in time $O(n^2 + dk^{d-1}(\log n)^d N)$ and space $O(n \log n + R)$ for almost random text databases.*

Without sampling. Finally, we see that in the case without sampling, i.e., $S = T$. In this case, Phase A is done in $O(n) = O(N)$ time by solving the color-set size problem in linear time [12] and by using the (compacted) GST directly as the AC-machine for L_1. Hence, we can show that the modified version solves the frequent pattern problem in time $O(dk^{d-1}(\log n)^d N)$ for almost random T, which improves the time complexity of the Split-Merge algorithm [6] by $\log n$ factor.

5 Experimental Results

We implemented the Levelwise-Scan algorithm in C++ and ran experiments on Reuters newswires [17]. The timing has been measured on a Sun workstation (300MHz UltraSPARC-II with 512MB main memory) under Solaris 2.6. As heuristics for finding interesting patterns in English texts, we set each sentence terminating with the period "." to be a document, and also removed from L_1 those phrases with frequency more than a user-specified threshold *max-freq* $= 0.5$ as *stop words*. To collect basic data, we put $S = T$ in the following experiments.

Performance overview. First, we run the experiments with varying the minimum support *min-freq* and the proximity k with the fixed sized sample of $n = 466KB$ (4,000 documents) and using the unordered and ordered 2-phrase patterns (Table 1 (a) and (b)). The proximity is $k = \infty$ for unordered and $k = 10$ for ordered patterns, where ∞ means that there is no proximity constraint. Table 1 (a) indicates that the number of frequent unordered patterns are six to ten times larger than that of frequent ordered patterns for the same value of *min-freq*, while the running times are similar.

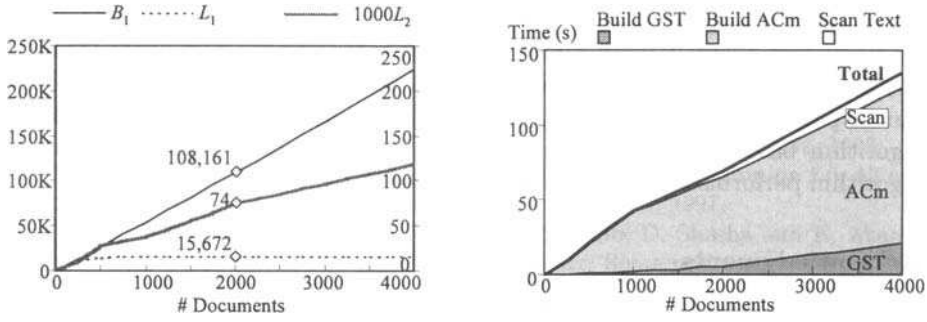


Fig. 6. (a) Characteristics of the text data: The number of basic features and frequent patterns with varying the number of documents from $26KB$ to $466KB$. B_1, L_1, and L_2 denote the total numbers of the branching phrases, $||L_1||$ and $||L_2||$, respectively. (b) Scalability: The running time with varying the number of documents from $26KB$ to $466KB$. Parameters are $k = 30, d = 2$, *min-freq* $= 0.005$, and *max-freq* $= 0.5$.

Scalability. Next, we made a scale-up test with varying the sample size $n =$ 25KB to 466KB (from 125 to 4000 documents) using ordered 2-phrase patterns, where still $S = T$. Fig. 6 (a) shows the behaviors of basic quantities B_1, L_1 and L_2 (explained in the caption of Fig. 6 (a)) that dominates the running time. We see that B_1 grows in $O(n)$ as expected, L_1 behaves like a huge constant, L_2 is small but slowly grows on this data. Fig. 6 (b) shows the total running time of Levelwise-Scan is almost linear in n, as expected from the theoretical analysis.

Simulation for mining a huge database with sampling. Finally, we estimated the expected running time of Levelwise-Scan with sampling from the experiments with text size up to $B = 466KB$ (Fig. 6) as follows. We used the formula $Time(N) = T_B^{\text{GST}} + T_B^{\text{ACm}} + (T_B^{\text{Scan}} + T_B^{\text{Trans}}) \times (N/B)$ to estimate the running time with sample size $n = B$ and the text database size N, where T_B^{α} denotes the time required for α-stage for $\alpha \in \{\text{GST, ACm, Scan}\}$, and $T_B^{\text{Trans}} = 0.05\text{s}$ is the transfer time for 1 MB from the disk in sequential I/O. Table 2 shows the estimated running time, where we see that Levelwise-scan will process a text database of 100MB under an hour with sampling ratio 0.5% on a corpus similar to the Reuters newswire.

Table 2. Expected running time on a large text database with a fixed sample size of $||S||$ = 466KB and a varying text database size $||T||$ up to 500MB. These running time are estimated by a simulation based on the result of Table 1.

Sample size	Text size	Sampling	Ordered Patterns		
$||S||$(MB)	$||T||$(MB)	ratio	GST+ACm (s)	Scan+Trans (s)	Total (s)
0.466	0.466	1.000	124.81	10.37	135.18
0.466	1.000	0.466	124.81	22.25	147.06
0.466	10.000	0.047	124.81	222.53	347.34
0.466	100.000	0.005	124.81	2225.32	40 min.
0.466	500.000	0.001	124.81	11126.61	3 hours

6 Conclusion

In this paper, we considered the discovery of unordered and ordered frequent phrase patterns from large text databases, and presented an efficient text mining algorithm based on the Apriori algorithm [3]. Experiments showed that this algorithm performs well on typical English text data.

Acknowledgments

We would like to thank to Prof. Shinichi Morishita, Prof. Takashi Tsuchiya, Prof. Masayuki Takeda, and Prof. Shinichi Shimozono for fruitful discussions and comments on this issue. This work is partially supported by a Grant-in-Aid for Scientific Research on Priority Areas "Discovery Science" from the Ministry of Education, Science, Sports, and Culture in Japan.

References

1. A. V. Aho, J. E. Hopcroft, and J. Ullman, *The design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, Fast discovery of association rules, Advances in Knowledge Discovery and Data Mining, Chap. 12, MIT Press, 307–328, 1996.
3. R. Agrawal, R. Srikant, Fast algorithms for mining association rules, In *Proc. the 20th VLDB*, 487–499, 1994.
4. A. V. Aho, M. J. Corasick, Efficient string matching: An aid to bibliographic search, In *CACM*, 1998
5. H. Arimura, A. Wataki, R. Fujino, S. Arikawa, An efficient algorithm for text data mining with optimal string patterns, In *Proc. ALT'98*, LNAI, 247–261, 1998.
6. H. Arimura, S. Shimozono, Maximizing agreement with a classification by bounded or unbounded number of associated words, In *Proc. ISAAC'98*, LNCS, 1998. A modified version is appeared as Efficient discovery of optimal word-association patterns in large text databases, *New Generation Computing*, 18, 49–60, 2000.
7. W. Croft, H. Turtle, D. Lewis, The use of phrases and structured queries in information retrieval. In *Proc. SIGIR'91*, 32–45, 1991.
8. L. Devroye, W. Szpankowski, B. Rais, A note on the height of the suffix trees, *SIAM J. Comput.*, 21, 48–53, 1992.
9. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds), Advances in knowledge discovery and data mining, AAAI Press/The MIT Press, 1996.
10. R. Feldman and W. Kloesgen, Maximal association rules: A new tool for mining for keyword co-occurrences in document collections, In *Proc. KDD-97*, 167–174, 1995.
11. T. Kasai, T. Itai, H. Arimura, Arikawa, Exploratory document browsing using optimized text data mining, In *Proc. Data Mining Workshop*, 24–30, 1999 (In Japanese).
12. L. C. K. Lui, Color set size problem with applications to string matching. Proc. the 3rd Annual Symp. Combinatorial Pattern Matching, 1992.
13. M. J. Kearns, R. E. Shapire, L. M. Sellie, Toward efficient agnostic learning. *Machine Learning*, 17, 115–141, 1994.

14. H. Mannila and H. Toivonen, Discovering generalized episodes using minimal occurrences, In Proc. KDD-96, 146–151, 1996.
15. E. M. McCreight, A space-economical suffix tree construction algorithm, In *JACM 23*, 262–272, 1976
16. S. Morishita, On classification and regression, Proc. DS'98, LNAI 1532, 1998.
17. D. Lewis, Reuters-21578 text categorization test collection, Distribution 1.0, AT&T Labs-Research, `http://www.research.att.com/~lewis/`, 1997.
18. J. T. L. Wang, G. W. Chirn, T. G. Marr, B. Shapiro, D. Shasha and K. Zhang, Combinatorial pattern discovery for scientific data: Some preliminary results, In *Proc. SIGMOD'94*, 115–125, 1994.

Using *Random Walks* for Mining Web Document Associations

K. Selçuk Candan[2,*] and Wen-Syan Li[1]

[1] C&C Research Laboratories, NEC USA, Inc., MS/SJ10, San Jose, CA 95134,USA
{candan,wen}@ccrl.sj.nec.com
[2] Computer Sci, and Eng. Dept., Arizona State University, Tempe, AZ 85287, USA
candan@asu.edu

Abstract. World Wide Web has emerged as a primary means for storing and structuring information. In this paper, we present a framework for mining implicit associations among Web documents. We focus on the following problem: "For a given set of seed URLs, find a list of Web pages which reflect the association among these seeds." In the proposed framework, associations of two documents are induced by the connectivity and linking path length. Based on this framework, we have developed a *random walk*-based Web mining technique and validated it by experiments on real Web data. In this paper, we also discuss the extension of the algorithm for considering document contents.

1 Introduction

In traditional information retrieval field, in order to determine the association between a given set of documents, keyword vectors that represent the contents of these document are compared. A major difference between Web pages and textual documents is that Web pages have links connecting to other related pages. When an author prepares a Web document, he/she would put contents on each page while linking related information together using anchors, to create a document spanning multiple pages. Thus, Web structures can be used as hints to derive document association. Existing approaches for finding Web document associations include the *companion* and *co-citation* algorithms proposed by Dean and Henzinger[1] and the Netscape algorithm[2] used to implement the `What's Related?` functionalities. In this paper, we are interested not only in deriving document associations, but also in inducing the reasons why they are associated. We focus on the problem "for a given two seed URLs, find a list of Web pages which reflex the association such two seed URLs."

Example 1. In Figure 1, we show a subset of links between two personal Web pages `W.Li` and `D.Agrawal`. The purpose of each link is indicated in the link label. The associations between `W.Li` and `D.Agrawal` are implicitly expressed in the Web structure connecting `W.Li` and `D.Agrawal` even though this structure is created independently by many individuals. Below, we enumerate some associations that can be derived from this graph, and some possible interpretations:

* This work was performed when the author visited NEC, CCRL.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 294–305, 2000.
© Springer-Verlag Berlin Heidelberg 2000

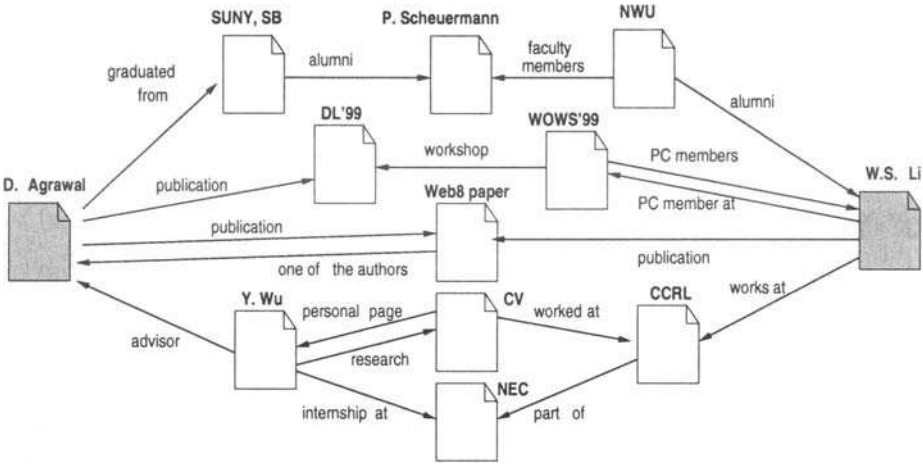


Fig. 1. Link structure connecting and associating the Web pages of `W.Li` and `D.Agrawal`

- `Web8 paper` page appears in a path of distance of 2 connecting the two pages. Therefore, `W.Li` and `D.Agrawal` may be associated due to a co-authored paper.
- `Y.Wu` page is on two paths each of distance 4. `W.Li` and `D.Agrawal` may be associated due to the fact they both supervised `Y.Wu`.
- `WOWS'99` and `ACM DL'99` pages appear on a single path of distance 3. `W.Li` and `D.Agrawal` are participating in the same conference.
- `P.Scheuermann`, `D.Agrawal`, `NWU`, `W.Li`, and `SUNY` pages appear on a single path of distance 4. `D.Agrawal` and `W.Li` may be associated due to some people related to `SUNY, SB` or due to an alumni relationship.

Obviously, the links connecting these pages are not intended to express such associations and the authors are not coordinated to make the link semantics consistent across the Web. However, we can use the following two intuitions: (1) *Path length:* Pages on a shorter path between the two pages in the example are stronger indicators than others to reflect why these pages are associated; and *Connectivity:* Pages which appear on more paths are stronger indicators than others to reflect why the two pages are associated.

Note that a page with a higher connectivity (i.e. more incoming links and outgoing links) is more likely to be included in more paths; consequently, such a page is more likely to be ranked higher according to the above criteria. This is consistent with the principle of topic distillation[3,4]. On the other hand, to address the associativity problem, we also need to consider the distance between a page and seed URLs to account for the first intuition. Thus, a page with a high connectivity but far away from the seed URLs may be less significant to represent the seed URLs after associations than a page with low connectivity but close to the seed URLs. A page which satisfies both criteria (i.e. near seed URLs and with high connectivity) is a good representative for the association.

Based on the motivation and intuitions, we present a novel framework for mining associations among Web documents using information *implicitly* reflected

in the links connecting them. We develop a Web mining technique, based on a *random walk algorithm*, which considers document distances by link and connectivity. In the concluding remarks, we also briefly show the algorithm can be extended to be specific content focused (e.g. finding why the `W.Li` and `D.Agrawal` are associated with respect to the *PowerBookmarks* project).

2 *Random Walks* Algorithm

In this section, we introduce the modeling of the framework and the algorithm.

2.1 Modeling

Let us assume that we are interested in mining the associations of a set, $\mathcal{S} = \{s_1, \ldots, s_n\}$, of seed Web pages (or *snodes*). The mining task is to find a set $Ref(\mathcal{S})$, of pages that best induce (or reflect) the association among a given set of *snode*s. We denote such pages inductive Web pages (or *inodes*). For ease of presentation, we start with the case where there are only two seed pages for association mining. The cases where $\mathcal{S}$ contains more than two pages is discussed in Section 2.3.

Let the Web be modeled as a directed graph, $G(V, E)$, and let the two seed pages in $\mathcal{S}$, defined as *snode*, correspond to vertices v_a and v_b in V. Let us also assume that we want to find an *inode* page (or vertex) within a radius of d from v_a or v_b. Note that the choice of d is application dependent. If *progressive* results are required, d can be incremented starting from 1, refining the results at each step, until either the process times out or an acceptable *inode* is located.

Links have been used in many fields to associate documents. They can be categorized into four types: connectivity, co-citation, social filtering, and transitivity[4]. Since the association mining problem as formulated in this paper is symmetric, we do not differentiate these. Consequently, we use an undirected graph, $G^u(V, E^u)$, to model the Web. Assuming that we are given a radius d, we define the relevant neighborhood, $G^N(V^N, E^N)$, of $G^u(V, E^u)$, as the set of vertices, $V^N = V_{G^u}(v_a, v_b, d)$, that are reachable either from v_a or v_b in d edge traversals: $(\forall v_i \in V_{G^u}(v_a, v_b, d)\ \ reachable_{G^u}(v_a, v_i, d) \vee reachable_{G^u}(v_b, v_i, d))$. Note that without loss of generality, we will assume that the graph, G^N, is connected.

To derive metrics for *Inode* selection, one straight forward candidate metric, that adjusts connectivity scores by distance, for *inode* selection would be

$$score(v) = \sum_{p \in paths(A,B,v)} \frac{1}{length(p)},$$

where $paths(A, B, v)$ is the set of (simple) paths between the seeds, A and B, that pass through a candidate *inode*, v, and $length(p)$ is the length of the path.

Note that, although it merges the two required structural criteria, this metric has two major disadvantages preventing its use in association mining. **(1)** First, its calculation may require the enumeration of all paths in the graph, which may require exponential time with respect to the size of the graph, and **(2)** although

1. $\mathcal{V} = \emptyset$;
2. For each $v_i \in V^N$, create a new node v'_i and insert it in $\mathcal{V}$;
3. $\mathcal{E} = \emptyset$;
4. For each $e_k = \langle v_i, v_j \rangle \in E^u$ such that both v_i and v_j are in V^N
   (a) create two directed edges $e'_{2\times k} = \langle v'_i, v'_j \rangle$ and $e'_{2\times k+1} = \langle v'_j, v'_i \rangle$ and insert them in $\mathcal{E}$;
5. For all vertices $v'_i \in \mathcal{V}$, let
   (a) $sdist(v'_i, v'_a)$ be the shortest distance, in G^N, between v'_i and the vertex, v'_a, corresponding to v_a, and
   (b) $sdist(v'_i, v'_b)$ be the shortest distance, in G^N, between v'_i and the vertex, v'_b, corresponding to v_b
   (c) $penalty(v'_i) = sdist(v'_i, v'_a) + sdist(v'_i, v'_b)$.
   (d) For all vertices $v'_i \in \mathcal{V}$ and for all $\langle v'_i, v'_j \rangle \notin \mathcal{E}$, $\mathcal{T}[j,i] = 0.0$;
   (e) For all vertices $v'_i \in \mathcal{V}$ solve the following set of linear equations:

$$L(v'_i) = \left\{ \sum_{\langle v'_i, v'_j \rangle \in \mathcal{E}} \mathcal{T}[j,i] = 1.0 \right\} \bigcup$$

$$\left\{ \mathcal{T}[j,i] \times penalty(v'_j) = \mathcal{T}[k,i] \times penalty(v'_k) \;\middle|\; \langle v'_i, v'_j \rangle \in \mathcal{E} \text{ and } \langle v'_i, v'_k \rangle \in \mathcal{E} \right\}$$

Fig. 2. Algorithm for constructing a random walk graph

the maximum length of the paths grows linearly with the number of vertices in the graph, the number of paths grows exponentially as shown in our experiments. As a consequence, contrary to the intuition, the effect of the long paths (since their number is exponentially higher than the number of shorter paths) on the calculation of $score(v)$ is likely to be much larger than the effect of short paths.

2.2 Case 1: $\mathcal{S}$ Contains Two Seed Pages

Consequently, instead of explicitly defining a metric, we will choose a set of random walk parameters that will implicitly capture the essence of these observations. For this purpose, we define and construct a random walk graph that reflects the required random walk parameters.

Definition 1 (Random Walk Graph). A random walk graph $\mathcal{R}(\mathcal{V}, \mathcal{E}, \mathcal{T})$ is a triple, where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ is a set of directed edges, and $\mathcal{T}$ is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix where

- $\mathcal{T}[j,i]$ denotes the likelihood of moving to vertex v_i from vertex v_j.

Note that $\sum_{1 \le j \le |\mathcal{V}|} \mathcal{T}[j,i] = 1.0$ o

Algorithm 2.1 (Constructing a Random Walk Graph) Given an undirected neighborhood graph $G^N(V^N, E^N)$, two vertices v_a and v_b in V, and a radius d, we can construct a directed random walk graph $\mathcal{R}_{(v_a, v_b, d)}(\mathcal{V}, \mathcal{E}, \mathcal{T})$ using the algorithm presented in Figure 2. o

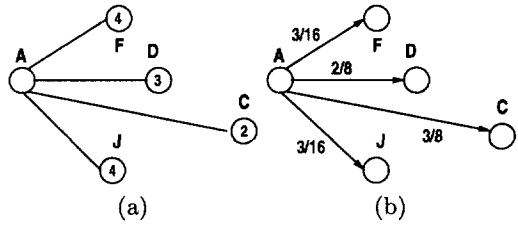


Fig. 3. (a) Penalty of each node and (b) transition values of each node

Description of the Algorithm for Constructing a Random Walk Graph Steps 1 and 2 of this algorithm insert the relevant vertices in the neighborhood into the random walk graph. Note that these two steps can be performed incrementally until a subgraph within a radius of d is explored. The next two steps use the undirected edges in the neighborhood graph to define two transitions (forward and backward) between the vertices in the random walk graph. These two transitions allow the random walk to proceed freely, back on forth, between the neighboring vertices of the graph.

Step 5, then, calculates a *penalty* for each node. This penalty term reflects the distance of each vertex from the seed vertices. Hence, for the case with two seeds, we define the penalty as the sum of shortest path distances between the given vertex and two seed vertices. We use the penalty to calculate the likelihood of each vertex being visited by the random walk process; more specifically, we calculate the transition probabilities of the edges in the graph using this term.

Since, by definition, a higher penalty means a greater distance from the seeds, it should yield a lower *association* score. Consequently, once the random walk process is at a vertex, v_i, it must proceed to a subsequent vertex, v_j, with a probability inversely proportional to v_js penalty. Furthermore, since the random walk will continue for an indefinite amount of time, the probability that the random walk process will leave vertex v_i (that is, it will proceed to one of its neighbors) must be equal to 1.0.

Example 2. Let us reconsider our example and focus on the portion of the graph shown in Figure 3(a), which depicts the vertex A, its four neighbors (F, D, C, and J), and the associated penalties calculated according to a distance metric (for the sake of simplicity, we omit the phase of penalty calculation). The following items reflect *some* of the facts about the transition probabilities of the edges leaving A:

- The sum of all such transition probabilities is equal to 1.0.
- Since the penalty of the vertex F is twice as much as the penalty of vertex C, the transition probability from A to F must be half of the transition probability from A to C.
- Since the penalty of the vertex D is $\frac{3}{2}$ times as much as the penalty of vertex C, the transition probability from A to D must be $\frac{2}{3}$ of the transition probability from A to C.

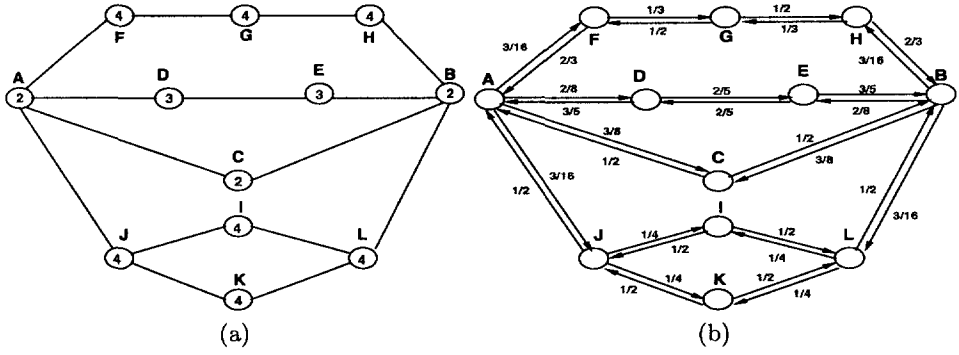


Fig. 4. (a) Penalty values, and (b) transition values for the example Web graph in Figure 1

Hence, we can calculate the transition values for the edges leaving A using the following set of constraints (as described in Step 5(e) of the algorithm):

$$\begin{aligned} \mathcal{T}[F,A] + \mathcal{T}[D,A] + \mathcal{T}[C,A] + \mathcal{T}[J,A] &= 1.0; \\ 4 \times \mathcal{T}[F,A] = 3 * \mathcal{T}[D,A]\ ; &\quad 3 \times \mathcal{T}[D,A] = 2 * \mathcal{T}[C,A]; \\ 2 \times \mathcal{T}[C,A] = 4 * \mathcal{T}[J,A]\ ; &\quad 4 \times \mathcal{T}[J,A] = 4 * \mathcal{T}[F,A]; \\ 4 \times \mathcal{T}[F,A] = 2 * \mathcal{T}[C,A]\ ; &\quad 4 \times \mathcal{T}[J,A] = 3 * \mathcal{T}[D,A]; \end{aligned}$$

Note that only the first four equations are enough to solve for all the unknowns. Figure 3(b) shows the transition values obtained by solving these constraints. o

Definition 2 (Convergence Vector). Given a random walk graph, $\mathcal{R}_{(v_a,v_b,d)}(\mathcal{V},\mathcal{E},\mathcal{T})$, t is called a convergence vector of $\mathcal{T}$ if $(t = \mathcal{T}t)$. o

Note that due to the structure of the transition matrix, such a convergence vector is guaranteed to exist. Intuitively, $t[i]$, describes the percentage of its time that a random walk process will spend in vertex $v[i]$ in a sufficiently long random walk. As we described earlier, *the higher this ratio, the better inode is the corresponding vertex. Consequently, we choose the inodes using their corresponding values in the convergence vector.*

Definition 3 (*Inode* Vertex). Given a graph $G(V,E)$, the *inode vertex* with respect to vertices v_a and v_b in G and a distance d, denoted as $inode_G(v_a,v_b,d)$ is a vertex $v_k \in V^N$ such that $t[k] = max\{t[i] \mid v'_i \in \mathcal{V}\}$. We also say that, if $t[i] > t[j]$, then v_i is more dominant than v_j. o

Example 3. Let us assume that Figure 4(a) shows a portion of a graph, G^u, where each shown vertex, v_i, is reachable from vertex A or B in 2 edges. The numbers shown in the vertices of the graph in Figure 4(a) are the corresponding distance penalties of the vertices. Figure 4(b), then, shows the corresponding random walk graph, $\mathcal{R}_{(A,B,2)}$. The transition values are shown as labels of the edges. The corresponding transition matrix $\mathcal{T}$ is also shown in Table 1(a).

Then, if we solve the linear equation $(I - \mathcal{T})t = 0$ (i.e. 12 variables and 13 constraints), we can find t as shown in Table 1(b). According to this, excluding

Table 1. (a) $\mathcal{T}$ and (b) t for for the example Web graph in Figure 1

$\mathcal{T}$	A	B	C	D	E	F	G	H	I	J	K	L
A	0.0	0.0	$\frac{1}{2}$	$\frac{3}{5}$	0.0	$\frac{2}{3}$	0.0	0.0	0.0	$\frac{1}{2}$	0.0	0.0
B	0.0	0.0	$\frac{1}{2}$	0.0	$\frac{3}{5}$	0.0	0.0	$\frac{2}{3}$	0.0	0.0	0.0	0.0
C	$\frac{3}{8}$	$\frac{3}{8}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
D	$\frac{2}{8}$	0.0	0.0	0.0	$\frac{2}{5}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
E	0.0	$\frac{2}{8}$	0.0	$\frac{2}{5}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
F	$\frac{3}{16}$	0.0	0.0	0.0	0.0	0.0	$\frac{1}{2}$	0.0	0.0	0.0	0.0	0.0
G	0.0	0.0	0.0	0.0	0.0	$\frac{1}{3}$	0.0	$\frac{1}{3}$	0.0	0.0	0.0	0.0
H	0.0	$\frac{3}{16}$	0.0	0.0	0.0	0.0	$\frac{1}{2}$	0.0	0.0	0.0	0.0	0.0
I	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	$\frac{1}{4}$	0.0	$\frac{1}{4}$
J	$\frac{3}{16}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	$\frac{1}{2}$	0.0	$\frac{1}{2}$	0.0
K	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	$\frac{1}{4}$	0.0	$\frac{1}{4}$
L	0.0	$\frac{3}{16}$	0.0	0.0	0.0	0.0	0.0	0.0	$\frac{1}{2}$	0.0	$\frac{1}{2}$	0.0

(a)

t	
A	0.183
B	0.183
C	0.137
D	0.076
E	0.076
F	0.051
G	0.034
H	0.051
I	0.034
J	0.068
K	0.034
L	0.068

(b)

the vertices A and B themselves, the most dominant vertex is C. Vertices, D and E follow C with lower dominance values as they are on a longer path between A and B. Although vertices J and L, are on an even longer path, they follow D and E closely since they are on multiple paths. o

2.3 Case 2: $\mathcal{S}$ Contains More Than Two Pages

In order to extend the algorithm presented in the previous section to the case in which $\mathcal{S}$ contains more than two seed pages, we need to observe that the algorithm uses these seed pages to discover the boundaries of the neighborhood, and to calculate the *penalty* of each vertex in the random walk graph.

The first of these tasks is easy to generalize. Given a set of vertices, $|\mathcal{S}| \geq 2$ and a radius, d, the relevant neighborhood, $G^N(V^N, E^N)$ of $G^u(V, E^u)$, is the set of vertices, $V^N = V_{G^u}(\mathcal{S}, d)$, that are reachable from the vertices in I in d edge traversals: $\left(\forall v_i \in V_{G^u}(\mathcal{S}, d) \quad \bigvee_{v_j \in \mathcal{S}} reachable_{G^u}(v_j, v_i, d)\right)$.

The second task, determining the penalty each vertex, can be handled in two ways. We can either trivially generalize the definition of the penalty as $\left(penalty(v'_i) = \sum_{v'_j \in \mathcal{S}} sdist(v'_i, v'_j)\right)$ or we can use

$$length(minimum\_steiner\_tree(\mathcal{S} \cup \{v'_i\}))$$

to get a more accurate picture of the distance of v'_i from the seed vertices. Note that the problem of finding the minimum weighted connected subgraph, G', of a given graph G, such that G' includes all vertices in a given subset R of G is known as the *Steiner tree* problem[1] [5]. Unfortunately, the minimum weight Steiner tree problem [6] is known to be NP-hard; i.e., it is not known whether there exists a polynomial time solution. The first option, on the other hand, is known to require polynomial time; and consequently, it is more efficient.

[1] If it exists, G' is guaranteed to be a tree.

2.4 Complexity of the Algorithm

For a given graph $G^N(V^N, E^N)$ the maximum degree of vertices is m, where $0 \leq m \leq |V^N|$, we analyze the complexity of the algorithm as follows:

1. The algorithm firsts find the shortest distance between every vertex and the seed vertices in $\mathcal{S}$. This operation can be performed using the Floyd-Warshall all pairs shortest path algorithm in $O(|V^N|^3)$. The assignment of penalties for each vertex, then, takes $O(|V^N|)$ time.
2. Once the penalties are know, calculation of the transition values for a given vertex takes $O(\mathcal{L}(m, m+1))$ time, where $\mathcal{L}(x, y)$ is the time that is required to solve a set of linear equations with x number of variables and y number of equations. Hence, the total number of time required to find all transition values in the graph is $O(|V^N| \times \mathcal{L}(m, m+1))$.
3. After all the transition values are known, the algorithm solves a set of linear equations with $|V^N|$ variables and $|V^N|+1$ equations. Hence, this step takes $O(\mathcal{L}(|V^N|, |V^N|+1))$.
4. Consequently, the total amount of time taken by the algorithm is

$$O(|V^N|^3 + |V^N| \times \mathcal{L}(m, m+1) + \mathcal{L}(|V^N|, |V^N|+1)).$$

3 Experiments

Our current implementation utilizes a linear equation solver `Maple` for calculating both edge transition probabilities and the corresponding convergence vector. We have conducted our experiments on *www-db.standford.edu*, which has 3600 pages and 12,581 edges. The average number of edges per page is 3.5. The experiments were ran on a 500MHz Pentium Architecture Linux OS PC with 128 MB of RAM.

3.1 Execution Time

The first experiment is for measuring the execution time of the algorithm. Note that there are two phases: calculation of **(1)** the edge transition probabilities and of **(2)**the convergence vector. We measure their execution time separately.

For a Web subgraph containing neighborhood of 1085 nodes (i.e. pages within 3 links from seed URLs `/classes.html` and `~echang/`, the execution time is reasonable fast. The total clock time needed is 760 seconds for the first phase; among that 572.76 seconds (i.e. 75.36%) is for reading the Web graph from disk. And, the clock time needed for the second phase is 343 second, among that only 9 seconds are for writing the results.

Thus, the total CPU time needed is 187.24 seconds for the first phase to solve 1,085 sets of equations (one for each node with average 4 to 5 variables and 4 to 5 constraints). In the second phase, only one set of equations needs to be solve. This set of equations with 1,085 variables and 1,086 constraints requires 334.36 seconds to solved. We believe such execution time is satisfactory and can be further improved by using a faster machine with larger memory or passing the results of the first phase directly as the input of the second phase.

Table 2. Top 10 association pages within radius of 1 from seeds

0.217939 /lore/	0.080059 /midas/midas.html
0.071164 /projects.html	0.056338 /c3/c3.html
0.053373 /~hector/infolab.html	0.053373 /projects-noim.html
0.053373 /projects-noim.html	0.048925 /tsimmis/tsimmis.html
0.040030 /tsimmis/	0.035582 /~chaw/

Table 3. 30 association pages within radius of 2 from seeds

0.066801 /lore/	0.015541 /tsimmis/tsimmis.html
0.049419 /www-db.stanford.edu/	0.015337 /~widom/
0.040217 /projects.html	0.014814 /c3/
0.038854 /~hector/infolab.html	0.014519 /people/gio.html
0.032719 /tsimmis/tsimmis.html	0.014315 /LIC/
0.028629 /projects-noim.html	0.013360 /~chaw/
0.028629 /projects-noim.html	0.013088 /~sergey/
0.028356 /~ullman/pub/hits.html	0.012270 /CHAIMS/
0.025630 /tsimmis/	0.012065 /people/hector.html
0.025175 /c3/c3.html	0.011043 /people/jpgs/
0.024539 /midas/midas.html	0.010225 /SKC/
0.022085 /people/widom.html	0.009979 /people/index.html
0.021813 /~widom/widom.html	0.009611 /people/index2.html
0.017109 /warehousing/warehouse.html	0.009161 /~ullman/
0.015678 /people/sergey.html	0.008930 /~tlahiri/

3.2 Association Mining Results

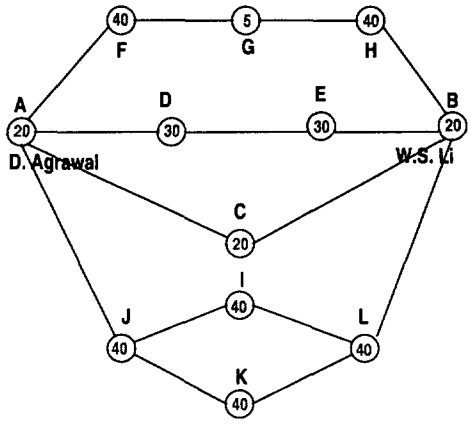
The second experiment is for testing the effectiveness of the algorithm. We can present only a small portion of the results due to the space limit.

For the project home page URLs `/lore/` and `/midas/midas.html`, the top 10 of 32 association pages within radius of 1 from seeds is shown in Table 2. We are satisfied with the results given most people working for both projects, such as `/~widom`, `/~ullman`, `/~wiener/`, and `/~sergey/`, as well as the home pages of related projects are selected. When we extend the exploration radius from 1 to 2, the results are still satisfactory given most of the results remain and few new pages are introduced. The associating pages within radius of 2 from seed URLs are sown in Table 3 (top 30 out of 155 pages are shown).

We observe that when the radius is extended, many index pages, such as the root page, departmental page (e.g. `infolab.html`, and more project pages (e.g. /warehousing/warehouse.html and /CHAIMS/), are now included. We also observe several other interesting factors. For example, Professor Wiederhold, whose home page is `/people/gio.html`, does not participate in LORE project and MIDAS project. However, Professor Wiederhold is the organizer of a popular database seminar where most people has links pointing to the seminar announcement page. As a result, the home page of Professor Wiederhold is selected.

4 Extension to Content-Focused Algorithm

In order to incorporate document contents to the association mining process, we propose to change the definition of penalty to also include document contents.



t	
A	0.123
B	0.123
C	0.092
D	0.051
E	0.051
F	0.116
G	0.185
H	0.116
I	0.023
J	0.046
K	0.023
L	0.046

(b)

Fig. 5. (a) Penalty of an example Web sub-graph, and (b) corresponding t

This variation, the *Content-Focused Random Walk Algorithm* , allows us to mine document associations with respect to not only seed URLs but also a particular topic. For example, we may ask a more specific question: "find why the pages `W.Li` and `D.Agrawal` are associated with respect to NEC" or "find why the pages `W.Li` and `D.Agrawal` are associated with respect to the `Y.Wu` page". Assuming that there exists a function, $relevance(v, topic)$, for measuring the relevance between the contents of a vertex v and a given topic *topic*, we redefine the penalty of a vertex as $\left(\frac{penalty(v)}{relevance(v,topic)}\right)$.

Alternatively, we can also consider content similarity of two linked pages to adjust the *importance* of each link. Intuitively, if a page is *content-wise* more related to the seed pages, then it is more likely to explain the association between them; hence it should be assigned a lower penalty, increasing its likelihood of being visited during a random walk. We call this variation `Content-Sensitive Random Walk Algorithm`. Assuming that there exists a function, $relevance(v, \mathcal{S})$, which evaluates the relevance of the content of a given vertex $v \in V$ with respect to a set of vertices $\mathcal{S} \in 2^V$, we can redefine the penalty of a vertex as $\left(\frac{penalty(v)}{relevance(v,\mathcal{S})+1}\right)$, $\left(\frac{penalty(v)}{relevance(v,\mathcal{S})}\right)$, or as $(penalty(v) \times (2 - relevance(v, \mathcal{S})))$. Note that the choice of the adjusted penalty function is application dependent and such a choice allows users to fine tune the sensitivity to the contents.

Example 4. Let us look reconsider the previous example. Now we want to ask "find why the pages `W.Li` and `D.Agrawal` are associated with respect to `Peter Scheuermann`. Let us assume that the page G has a strong relevance to the focused content, "Peter Scheuermann". Let us also assume that the relevance function used assigns 0.8 to G and 0.1 to all other pages.

Assuming that we use the penalty function $\left(\frac{penalty(v)}{relevance(v,I)}\right)$, Figure 5 shows the graph and the corresponding convergence matrix, t. According to this vector, the most "dominant" vertex in the graph is G. Comparing with the results in

Table 1(b), the scores of G, F, and H are boosted since G is now in focus. In this example, we observe that the results successfully reflect the structure and the document contents with respect to the given topic. o

5 Related Work

Link information has been used by many search engines to rank query results. They assume that the quality of a document can be "assured" by the number of links pointing to it. HITS algorithm was proposed by J. Kleinberg [3]. It aims at selecting a small subset of the most "authoritative" pages from a much larger set of query result pages. Authoritative page is a page with many incoming links and a hub page is a page with many outgoing links. Such authoritative pages and hub pages are mutually reinforced: good authoritative pages are linked by a large number of good hub pages and vice versa. This technique organizes topic spaces as a smaller set of hub and authoritative pages and it provides an effective mean for summarizing query results, so called "topic distillation".

Bharat and Henzinger [7] improved the basic HITS algorithm [8] by adding additional heuristics. The modified topic distillation algorithm considers only those pages that are in different domains with similar contents for mutual authority/hub reinforcement. Another variation of the basic topic distillation algorithm is proposed by Page and Brin[9]. Their algorithm further considers page fanout in propagating scores.

Many of these basic and modified topic distillation algorithms have been also used to identify latent Web communities[10,11]. These above techniques focus on finding high quality documents induced by link analysis. Our proposed algorithm can be extended for topic distillation by targeting all nodes in the explored graph instead of few seed URLs. By such adjustment, the algorithm would be able to find hub and authority.

Dean and Henzinger[1] proposed two algorithms, *companion* and *cocitation* to identify related pages and compared their algorithms with the Netscape algorithm[2] used to implement the `What's Related?` functionalities. By extending the scope from documents to Web sites, Bharat and Broder[12] conducted a study to compare several algorithms for identifying mirrored hosts on the Web. The algorithms operate on the basis of URL strings and linkage data: the type of information easily available from web proxies and crawlers.

This work above focuses on finding related documents or Web sites. Our work focuses on finding pages inducing associations of given seed URLs. The proposed random walks algorithm can be extended to be content-focused.

6 Concluding Remarks

In this paper, we present a framework and an algorithm for mining implicit associations among Web documents induced by link structures and document contents. The algorithm works on any graph and can focus on a specific topic. We have implemented and evaluated the algorithm. The preliminary experimental results on real Web data show that the algorithm work well and efficiently.

The authors would like to express their appreciations to `www-db.stanford.edu` for its data used in their experiments. Selecting this Web site is due to the considerations (1) the authors need to be familiar with the contents so that the authors can evaluate the results; and (2) the pages in the Web sites must not be dynamically generated pages. The second consideration restricts the authors from using most of corporation sites. The experimental results presented in this paper are for the purposes of scientific research only.

References

1. Jeffrey Dean and Monika Henzinger. Finding Related Pages in the World Wide Web. In *Proceedings of the 8th World-Wide Web Conference*, Toronto, Canada, May 1999.
2. Netscape Communications Corporation. What's Related web page. *Information available at http://home.netscape.com/netscapes/related/faq.html.*
3. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, January 1998.
4. Wen-Syan Li and Selcuk Candan. Integrating Content Search with Structure Analysis for Hypermedia Retrieval and Management. To appear in *ACM Computing Survey*, 2000.
5. Frank K. Hwang, Dana S. Richards, and Pawel Winter, editors. *The Steiner Tree Problem (Annals of Discrete Mathematics, Vol 53).* 1992.
6. S.L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1:113–131, 1971.
7. Krishna Bharat and Monika Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21th Annual International ACM SIGIR Conference*, pages 104–111, Melbourne, Australia, August 1998.
8. Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. In *Proceedings of the 7th World-Wide Web Conference*, pages 65–74, Brisbane, Queensland, Australia, April 1998.
9. Lawrence Page and Sergey Brin. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the 7th World-Wide Web Conference*, Brisbane, Queensland, Australia, April 1998.
10. David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Inferring Web Communities from Link Topology. In *Proceedings of the 1998 ACM Hypertext Conference*, pages 225–234, Pittsburgh, PA, USA, June 1998.
11. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the Web for Emerging Cyber-Communities. In *Proceedings of the 8th World-Wide Web Conference*, Toronto, Canada, May 1999.
12. Krishna Bharat and Andrei Z. Broder. Mirror, Mirror, on the Web: A Study of Host Pairs with Replicated Content. In *Proceedings of the 8th World-Wide Web Conference*, Toronto, Canada, May 1999.

A Concurrent Approach to the Key-Preserving Attribute-Oriented Induction Method

Maybin K. Muyeba and John A. Keane

Department of Computation, UMIST, Manchester M60 1QD, UK.
{muyeba, jak@co.umist.ac.uk}

Abstract. Attribute-Oriented Induction (AOI) reduces the search space of large data to produce a minimal rule set. Classical AOI techniques only consider attributes that can be generalised but eliminates *keys* to relations. The Key-Preserving AOI (*AOI-KP*) preserves *keys* of the input relation and relate them to the rules for subsequent data queries. Previously, the sequential nature of AOI-KP affected performance on a single processor machine. More significantly, time was spent doing I/O to files linked to each generated rule. AOI-KP is $O(np)$ and storage requirement $O(n)$, where n and p represent the number of input and generalised tuples respectively. We present two enhanced *AOI-KP* algorithms, *concAOI-KP* (concurrent *AOI-KP*) and *onLineConcAOI-KP* of orders $O(np)$ and $O(n)$ respectively. The two algorithms have storage requirement $O(p)$ and $O(q)$, $q = p*r$, $0<r\leq 1$ respectively. A prototype support tool exists and initial results indicate substantially increased utilisation of a single processor.

1 Introduction

Data mining [2] is the application of algorithms to discover knowledge in data. Attribute-Oriented Induction (AOI) has been investigated for mining various kinds of rules including associations, sequential patterns, classification and summarisation [3]. AOI is a set-oriented generalisation technique that produces high-level rules from huge data sets. AOI reduces the input relation to a minimal relation called a *prime table* and then a *final rule table* by using an *attribute* or *rule threshold*. An *attribute* or *rule threshold* determines how any distinct attributes or rules remain in the *final rule table*. For each attribute, AOI uses a concept hierarchy tree [4] to generalise it by climbing through the hierarchy levels of that attribute. An attribute is generalised if its low-level concepts (e.g. leaf concepts) are replaced by high-level concepts. Database values are stored as leaf concepts in the tree (see figure 1).

In many approaches using AOI, the problem has been losing information upon generalisation. This happens when relational *keys* or attributes that index these relations are removed. This is because the index can not be generalised.

Keys preserved during the mining process can be used to query data relevant to the rules produced. For example, the rule in figure 2 discovers knowledge about postgraduate students in year one with respect to their gender and birth place. The rule

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 306-316, 2000.
© Springer-Verlag Berlin Heidelberg 2000

reveals the ratios of each category of students and their specific identities. The keys for each rule can be used to efficiently query data related to the three rules.

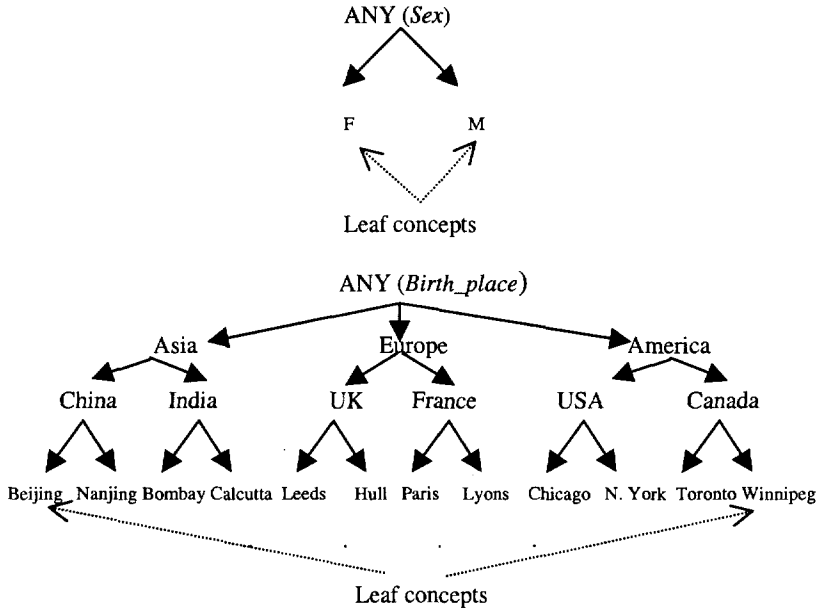


Fig. 1. Concept hierarchies for Sex and Birth_place

$\forall$(x) Postgraduate (x)$\wedge$Acc_year (x) ="Year 1"$\Rightarrow$
Birth_place (x)$\in$ America [30%][Rule *keys* =1, 3, 10]
$\vee$ Sex (x) = "Female" $\wedge$ Birth_place (x)$\in$ Europe [30%][Rule *keys* =2, 6,7]
$\vee$ Sex (x) = "Male"$\wedge$ Birth_place(x)$\in$ Asia [40%][Rule *keys* =4, 5, 8, 9]

Fig. 2. Characteristic rules for postgraduate students

The query "List the names and average marks obtained by female European postgraduate students in year 1" would be efficiently queried using *keys* 2, 6 and 7 from an appropriate table or tables storing this data. The query would be more efficient if the preserved *keys* index the queried tables. Thus, generalised information coupled with the ability to obtain detail data from the database is necessary for querying the mined data.

The Key-Preserving AOI algorithm *(AOI-KP)* [5] performs efficient data queries on the discovered knowledge by making use of *keys*. In AOI-KP, the input data is stored in an initial table while *keys* to tuples are later inserted in a separate table during the mining process. When static data structures are used for storing *keys* in

memory, performance is drastically affected. By using dynamic data structures that grow at execution time, there are gains in space and time as will be shown in section 3. Initial results indicate that using the AOI-KP approach helps to shuffle back and forth between rules produced and the database.

This paper introduces a concurrent key-preserving algorithm *conAOI-KP*, motivated by the performance degradation of the sequential AOI-KP algorithm as the volume of input data increases. The approach to concurrency is as follows:

1. The merging of tuples, to be illustrated in section 3, is a sequential process. This involves removing some tuples from the *prime table* and then inserting *key(s)* in a table of *keys*. These two tasks could therefore occur concurrently as they accessed different tables.
2. The I/O task for writing keys to file after rule production were not time efficient. After accumulating *keys* in a table, each rule was then associated with some *keys* in that table and had to be written to file for subsequent data queries. Thus, the greater the number of rules, which depend on the *rule threshold*, the more file writes were needed.

A major hindrance is storing the whole input to memory. A solution is to retrieve blocks of tuples to memory, generalise them and only store the generalised tuples before the next block is retrieved. Another algorithm, *onLineConAOI-KP*, is introduced to enhance *conAOI-KP* and is more space efficient but less time efficient.

The paper is organised as follows: in section 2, related work is considered; in section 3 analysis and results of the algorithms are presented; and section 4 presents conclusions and further work.

2 Related Work

This work is similar to the algorithms on AOI such as Learning CHaracteristic Rules (*LCHR*), and in particular to Generalise DataBase Relation (*GDBR*) and Fast Incremental Generalisation and Regeneralisation (*FIGR*) [6].

Generally, it is desirable that data mining algorithms be both space and time efficient. Like *GDBR*, *conAOI-KP* is transformed into an on-line algorithm (*onLineConcAOI-KP*) by reading blocks of tuples from disk to memory. The difference here is that the size of the *prime table* is determined from distinct non-leaf concepts, which are a level higher than leaf concepts, for each attribute. *GDBR* determines the size of the prime table from *attribute thresholds*. Therefore, it is assumed that every tuple will be generalised at least once so that there is no allowance for any tuple proceeding to the rule generation stage with leaf concepts. This is a remote possibility with *GDBR* if *thresholds* are reached before any generalisation takes place. However, attributes with only *two* concept *levels*, for example "sex" which has leaf concepts "M" and "F" and general concept "ANY" are the only exceptions. Similarly, *FIGR* determines the size of *prime table* from distinct values of leaf concepts.

3 The Concurrent AOI-KP Approach

3.1 Analysis of Order Complexity and Space Requirements

The motivation for this work was to improve the sequential AOI-KP's order complexity of $O(np)$ and space requirement of $O(n) + O(p)$, for n input and p generalised tuples.

Firstly, with the *onLineConAOI-KP* algorithm, n input tuples were retrieved as blocks of tuples with each tuple generalised before insertion in the initial relation. Subsequent tuples were retrieved, generalised and compared to other generalised tuples in memory. Similar tuples (those having the same attribute values except their *keys*) were stored as one tuple (this is termed *merging tuples*) and their *keys* stored in a table, called a *keys table*. Therefore the initial input table, which we now call a *prime table*, can be declared with only a few hundred rows. This is because the product of distinct values of each attribute's non-leaf concepts (i.e. the generalised concepts) is smaller than distinct values of leaf concepts.

The *prime table's* size is therefore a small constant value of size

$$q = p^*r,\ 0<r\leq 1.$$

Now $q = n$ if all input tuples were retrieved to memory. Considering FIGR and GDBR, q<p if GDBR with a big *attribute threshold* and large number of attributes is considered as GDBR determines the size of the *prime table* from the product of *attribute thresholds*. FIGR determines the size of the *prime table* from distinct leaf concepts, which are greater than distinct generalised leaf concepts as implemented in *onLineConAOI-KP* and *conAOI-KP*.

Therefore, the initial relation in our case is just a prime relation of size $q<=p$. This approach also eliminates the use of a summary input table as used in GDBR and FIGR. This summary table is presented as a two-dimensional table with two attributes relevant to the mining process. It compiles statistics about those attribute values and could be prohibitively large if the number of attributes increased.

With FIGR, the size of the *prime table* with m input attributes is

$$s = \prod_{i=1}^{m} k_i$$

where k_i is the distinct number of leaf concepts for each attribute i. For *onLineConAOI-KP*, the size of the *prime table* is

$$q = \prod_{i=1}^{m} k_{gen(i)}$$

where $q = p*r$, $0<r\leq 1$, as $k(gen_{(i)})$ is the distinct number of generalised leaf concepts for generalised attribute i and ofcourse $k(gen_{(i)}) < k_i$.

Therefore, *onLineConAOI-KP*'s space requirement is $O(q)$ for the initial and prime relations and $O(n/m)+c$ for the *keys table*, where c is a small storage increment due to key insertions. The $O(n/m)+c$ arises because *keys table* dynamically increases in size as more input is read. This is not the case when a fixed size *keys table* for n input tuples that requires $O(n)$ space is declared.

As more input is read, c gets used up and can grow dynamically in multiples of its previous size. This means for very large n, a single machine can still have memory problems unless m is large which is unlikely for the AOI method. Therefore, the space requirement for large n is $O(q)$ and order complexity $O(n)$ for *onLineConAOI-KP*.

Secondly, for *conAOI-KP*, the time to retrieve n input tuples, convert them to concepts for each of the m attributes to a *prime table* of size p is $O(nmpd)$, where d is the deepest concept hierarchy for any attribute. Assuming a small number of attributes m and the deepest concept hierarchy d is small, the order complexity is only $O(np)$.

The problem with *conAOI-KP* is that we do not retrieve blocks of tuples to memory but the whole input is read. The only improvement is concurrency of the file I/O process.

We present concurrency mechanism involved in both *conAOI-KP* and *onLineConAOI-KP* in the next section and compare performance.

3.2 Sequential and Concurrent Algorithms

In this section, the sequential and concurrent versions of the AOI-KP algorithm are discussed. Two class pseudo-codes and their methods for the parent and children thread processes are also shown for clarity. Three major processes where concurrency may be useful namely, *key insertion*, *merging similar tuples* and *file I/O* have been investigated.

The sequential *AOI-KP algorithm*, in figure 3, performs poorly with large data input because it uses memory to hold all the data. The two processes, tuple merges and key insertion, use separate tables for storing generalised tuples (the *input or prime table*) and *keys* (the *keys table*). A tuple merge occurs if two tuples have the same attribute values (generalised or not) except their identifying attribute called a *key*. Thus one tuple and its corresponding *key(s)* is removed from the *prime or input table* and the resulting two *keys* are inserted in the *keys table*. These two processes could therefore occur concurrently.

However, a much bigger difference in execution time occurs when there are a large number of *keys* associated with each of the two tuples being compared in the *prime table* and a merge has to be performed. Therefore, two threads of execution for

key insertion and *tuple merges* could be spawned concurrently. In theory, this means that with smaller data sizes, the concurrent approach would be much slower than the sequential approach because of *thread* overheads i.e. the time to create, spawn, synchronise and terminate threads.

In addition, however, a major performance bottleneck is writing *keys* to file when rules are generated. Therefore, to improve the overall performance of AOI-KP, the concurrent *AOI-KP (conAOI-KP)* algorithm is introduced as shown in figure 4.

```
Step 1.        Collect data (as a whole)
        Make concept hierarchies for Ai
      Determine distinct values of attributes Ai
               For each attribute Ai
                        While attribute threshold not reached
                          If ( Ai has hierarchy, Ai <> key attribute, Ai has more
                        than two levels)
                                  Generalise Ai
Step 2.        Merge similar tuples
               If two tuples are similar
                        Insert keys in Keys Table (copying key (s) )
               Delete tuple(s), leave one tuple in prime table (tuple merge)
                        Increase counts
      Else
                        Insert tuples in different rows of Prime Table
                        Insert keys in different rows of Keys Table
               Increase counts
Repeat Step 2 until attribute threshold is reached
Step 3. Check rule threshold
      While rule threshold not reached
                        Generalise appropriate attribute
               Merge tuples
                 Insert keys
      For i =1 to rule threshold (sequential I/O file process)
                        Search keys Table for non-Null row
               If (non-Null row Found)
                          Write keys to file iR.txt
//Sequential AOI-KP
```

Fig. 3. Sequential AOI-KP Algorithm

Figure 5 shows class implementations of the parent and child thread codes for concurrent execution of the *file I/O* process. The parent class passes to the child class method, the file number and the row of the *keys table* where the preserved *keys* are stored. The file number corresponds to the rule number produced in the mining process e.g. rule 1 has preserved *keys* in file 1R.txt etc.

```
Step 1. Collect data (In Blocks for on line concurrent AOI-KP)
        Make concept hierarchies for Ai
        Determine distinct values of attributes Ai
        For each attribute Ai
                While attribute threshold not reached
                If ( Ai has hierarchy, Ai <> key attribute, Ai has more than
                two levels)
                        Generalise Ai
Step 2. Merge similar tuples
        If two tuples are similar
                Perform concurrent Key insertion and tuple merges
                        Insert keys in Keys Table (copying key (s) )
                        Delete tuple(s), leave one tuple in prime table
                        (tuple merge)
                Increase counts
        Else
                Insert tuples in different rows of Prime Table
                Insert keys in different rows of Keys Table
                Increase counts
Repeat Step 2 until attribute threshold is reached
Step 3. Check rule threshold
        While rule threshold not reached
                Generalise appropriate attribute
                Merge tuples
                Insert keys
        For i =1 to rule threshold (concurrent I/O file process)
                Spawn parent thread when first non-null
                        or non-blank row of Keys Table found
                Spawn child thread to write keys to file iR.txt
                Search Keys Table for next non-Null row
                If (non-null row Found)
                   Spawn next child thread to write keys to file iR.txt
//Concurrent AOI-KP
```

Fig. 4. Concurrent AOI-KP Algorithm

With file I/O, once *keys* are stored in the *keys table*, each set of *keys* for a rule is separated by a null or blank row. To store *keys* for rule *i* in file *i*R.txt, the parent thread is spawned and scans the *keys table* for a blank or null row, having spawned a child thread from the first encountered non-null or non-blank row of the *keys table*. If another blank or null row is subsequently encountered, another child thread for rule *i+1* is spawned for writing to file (*i+1*)R.txt and so on.

If the *attribute* or *rule threshold* is *m*, then *m* children threads will be spawned for writing to *m key* files.

```
Class ParentThread
{  (Declare member variables and assign them values using
        constructors)
        ...
        i=1
        fileNo=1
        k=0
        ruleThreshold = m
        ...
        ChildThread ch1,..,chi, i =1..ruleThreshold
        ...
Method (k, ruleThreshold)
while ( k < size of Keys Table AND i<=ruleThreshold )
{
        if( blank or null value encountered in keys Table  row k) loop
                case (fileNo)
                        1:      spawn childThread ch1(fileNo, k)
                        2:      spawn childThread ch2(fileNo, k)
                        3:      ..
                         ...
                         ...
                        i:      spawn childThread chi(fileNo,k)
                        k=k+1
                        i=i+1
                        fileNo=fileNo+1
        }
}// end  Class ParentThread

Class ChildThread
{  (Declare member variables and assign them values using
        constructors)
        fileNo
        rowK
        for ( i  = 1,..,ruleThreshold)
                        Declare files iR.txt
        ...
Method ( fileNo, rowK)
case (fileNo)
        1:      write rowK of Keys Table in file 1R.txt
                continue;
          :
        m:      write rowK of Keys Table in file mR.txt
                continue;
        close files
}//end class childThread
```

Fig. 5. Classes of Parent and Child Threads for file I/O

Figure 6 shows the results of execution times of sequential *AOI-KP*, *conAOI-KP* and *onLineConAOI-KP* algorithms on student data using a P266 MHz 64MB-memory

Windows NT cluster. The execution times include data retrieval, mining and file I/O processes. Each data set was run five times and the average time computed.

The concurrent *AOI-KP* thus improved processor utilisation by 25% due to concurrent I/O for each file of *keys* generated.

The *onLineConAOI-KP* algorithm was run on the same data set. By retrieving blocks of tuples to memory, generalising the tuples and inserting them in a prime table directly from the input, memory restrictions were temporarily solved. As more input was read, the *keys table* grew with the input size. As database retrieval time is an expensive operation, this affected execution time, which improved processor utilisation by only 16% compared to the sequential *AOI-KP* algorithm.

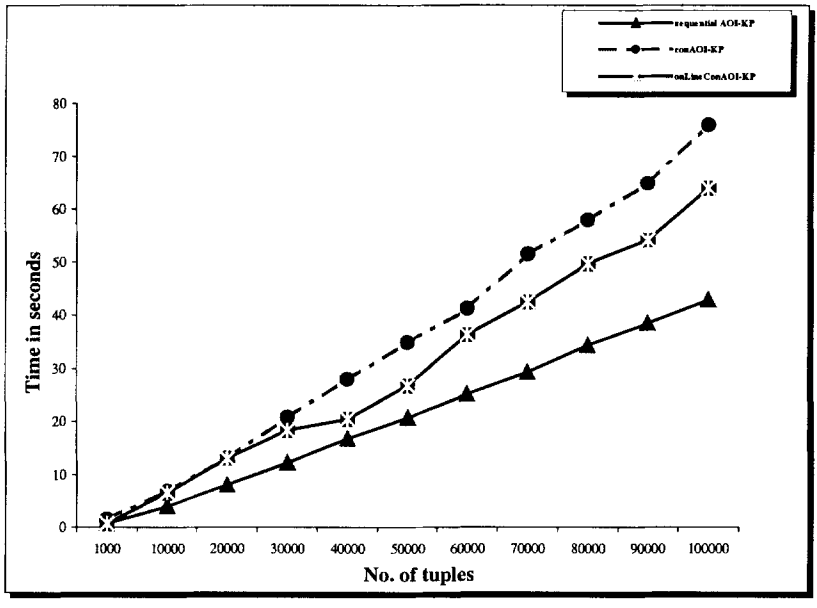


Fig. 6. Comparisons of sequential and concurrent versions of AOI-KP

From the graph in figure 6, it is evident that concurrency mechanisms have provided improvement. All three algorithms are affected by the size of input data. The *onLineConAOI-KP* algorithm can take more input than the other two because memory is economised by retrieving blocks of tuples. The *conAOI-KP* algorithm runs faster than both sequential *AOI-KP* and *onLineConAOI-KP* as long as the data fits in memory. This is because it is exempt from the I/O bottleneck experienced by *onLineConAOI-KP* and employs concurrency not implemented in *sequential AOI-KP*. However, sequential *AOI-KP* initially performs equally well with few input data. As memory gets used and file I/O starts, performance deteriorates.

Further improvements would be to lessen use of memory by retrieving blocks of tuples, writing keys to file immediately upon data input or employing a parallel-processing paradigm. The former would be more space efficient but not time efficient, whilst the latter approach may improve both space and time efficiency.

4 Conclusions and Further Work

This paper has presented a concurrent approach for preserving *keys* in attribute-oriented induction on a single processor machine. The result has been enhanced performance. The execution time, when *conAOI-KP* is considered, has been reduced by 25%, indicating a greater utilisation of the processor. In addition, the storage requirement is only *O (p)*. When *onLineConAOI-KP* is considered, execution time is reduced by 16% and only required *O (q)* space.

With large data, the *keys table* grows proportional to the input. In addition, database retrieval time is affected by large inputs especially when blocks of tuples have to be read. This was shown by the deterioration in execution time of the *onLineConAOI-KP* algorithm with respect to *conAOI-KP*.

Another viable solution is to employ parallelism in either of the following two ways:

1. Employing a distributed memory message-passing architecture e.g. a network of workstations (NOW). A NOW provides attractive scalability in terms of computational power and memory[7].
2. Employing a parallel shared memory multiprocessor with explicit message-passing [8].

The amount of execution time would be greatly reduced if *h* processors were involved. This would mean reducing the order complexity from *O (np)* to *O (np/h)* for the *conAOI-KP* algorithm and *O (n)* to *O (n/h)* for the *onLineConAOI-KP* algorithm. Current work investigates parallelism on a NOW architecture.

References

1. Han, J.; Cercone, N. and Cai, Y. 1991 "*Attribute-Oriented Induction in Relational Databases*" In G. Piatetsky-Shapiro and W. J. Frawley, editors, Knowledge Discovery in Databases, pp 213-228.
2. Frawley, W. J. and Piatetsky-Shapiro, G. 1991. "*Knowledge Discovery in Databases*", AAAI/MIT Press, pp 1-27.
3. Agrawal, R.; Imielinski, T. and Swami, A. 1993. "*Database Mining: A Performance Perspective*" IEEE Transactions on Knowledge and Data Engineering, 5(6):914-925, December.
4. Fu, Y. 1996. "*Discovery of Multiple-Level Rules from Large Databases*" Ph.D. thesis, Computing Science, Simon Fraser University, July.
5. Muyeba, K. M. and Keane, J. A. 1999. "*Extending Attribute-Oriented Induction as a Key-Preserving Data Mining Method*" Proceedings of the 3rd European Conference on Principles of Knowledge Discovery and Data Mining (PKDD'99),

J. Zytkow and J. Rauch, editors, Prague, Czech. Republic, pp 249-258, September.

6. Carter, C. L. and Hamilton, H. J. 1998. "*Efficient Attribute-Oriented Generalisation for Knowledge Discovery from Large Databases*" IEEE Transactions on Knowledge Discovery and Data Engineering, 10(2):193-208, March.
7. Zaki, M. J.; Li, W. and Parthasarathy, S. 1997. "*Customized Dynamic Load Balancing for a Network of Workstations*" JPDC, Special Issues on Workstation Clusters and Network-Based Computing, 43(2):156-162, June.
8. Freitas, A. A. and Lavington, S. H. 1996. "*Mining Very Large Databases with Parallel Processing*", Kluwer Academic, Boston.

Scaling Up a Boosting-Based Learner via Adaptive Sampling

Carlos Domingo * and Osamu Watanabe **

Dept. of Math. and Comp. Science, Tokyo Institute of Technology
Meguro-ku, Ookayama, Tokyo, Japan.
{carlos,watanabe}@is.titech.ac.jp

Abstract. In this paper we present a experimental evaluation of a boosting based learning system and show that can be run efficiently over a large dataset. The system uses as base learner decision stumps, single atribute decision trees with only two terminal nodes. To select the best decision stump at each iteration we use an adaptive sampling method. As a boosting algorithm, we use a modification of AdaBoost that is suitable to be combined with a base learner that does not use all the dataset. We provide experimental evidence that our method is as accurate as the equivalent algorithm that uses all the dataset but much faster.

1 Introduction

One defining characteristic of data mining applications is that the input dataset is huge. Thus, we are typically seeking for algorithms that can be run efficiently even if the input is very large. Another important aspect of data mining is that, in many situations, one is required to obtain solutions that can be later interpreted by a human expert. In contrast, in machine learning research the emphasis has been traditionally put in prediction accuracy. Thus, state of the art algorithms like C4.5 that consistently produce very accurate hypothesis, when run over a large dataset are slow and produce very large outputs that are hard to interpret. It is a common practice for data miners to run an algorithm for induction of decision tress over a large dataset for several hours only to end up discarding most of the output (and thus, reducing the accuracy) in order to obtain something that can be interpreted by a human expert.

In this paper we present a learning algorithm that while not always producing a hypothesis as accurate as one might obtain using traditional machine learning methods, it produces a very concise output and can be run very fast even if the dataset is large. This learning algorithm combines two powerful tools, *boosting* and *sampling*. We will use a very simple learning algorithm that uses an adaptive sampling method as a base learner and then we will use a boosting method to

* Thanks to the European Commission for their generous support via a EU S&T fellowship programme.

** Supported in part by the Ministry of Education, Science, Sports and Culture of Japan, Grant-in-Aid for Scientific Research on Priority Areas (Discovery Science).

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 317–328, 2000.
© Springer-Verlag Berlin Heidelberg 2000

improve its accuracy. In the following we discuss these two methods in more detail.

Boosting [21] is a technique for constructing a "strong" learning algorithm based on a "weak" learning algorithm. Boosting typically works by repeatedly disturbing the training set to obtain several different weak hypotheses that concentrate more and more are on the harder instances and that are later combined to obtain a much stronger one. In particular, the AdaBoost algorithm of Freund and Schapire [11] has been repeatedly reported to be the most effective in the absence of noise [12,19,2,1]. However, AdaBoost is originally designed to run using all the dataset and thus it is not suitable to be used with large datasets. In this paper we will use a modification of AdaBoost recently proposed by Domingo and Watanabe [23,5] that is more suitable for being combined with a base learner that uses only a portion of the dataset selected through sampling as discussed in Section 2. The boosting algorithm outputs a hypothesis that is the weighted majority of the hypothesis output by the base learner.

Since as we argued one of our goals is to obtain a hypothesis that can be easily interpreted, we will use as a base learner a very simple one, *decision stumps* (single attribute decision trees with only two terminal nodes). Obviously, the predictive power of this learner is very weak and this is the reason we also want to use boosting to combine few of them to increase the accuracy. Thus, at each boosting iteration we will select the best stump with respect to the distribution generated by the boosting process at that step. Recall that we also wanted to be able to efficiently run the learning algorithm on a large dataset. Boosting is a slow method since it implies running sequentially several times the base learner algorithm. Thus, unless we are not able to make the base learner very fast, the overall learning algorithm might become extremely slow. The method we propose to speed up the base learner is sampling. Instead of using all the dataset at each iteration to determine which stump we pass to the boosting algorithm we will use only a portion of it. Now, the problem is shifted to decide how much amount of data we need at each iteration. To solve this problem we will use an *adaptive sampling* method proposed by Domingo et.al. [3,4] that it is particularly suitable for this problem. Adaptive sampling methods do not determine the sample size a priori. Instead, they obtain examples incrementally and decide on-line depending on the current situation when to stop sampling. Adaptive sampling methods have been studied in statistics under the name of sequential sampling [22] and more recently in the database community [17,16]. In the KDD literature, related methods are described under the name of progressive sampling [14,18]. Details on how to use sampling for selecting the stump at each boosting iteration are provided in Section 3.

In this paper we provide an experimental evaluation of this learning system. We have done experiments to compare the results obtained using all the dataset at each boosting iteration to select the best decision stump and the results obtained using our adaptive sampling method to do the selection. We conclude that there is no apparent lost in accuracy by using the sampling method while there is a great decrease in running time, an average of 40 times for the fastest

method over the 6 datasets used in the experiments. All the hypotheses obtained are just a weighted combination of 10 attributes.

This paper is organized as follows. In Section 2 and Section 3 we describe the boosting algorithm MadaBoost and the decision stump selector with adaptive sampling that we will use for our experiments. Then, in Section 4 we report several experiments that we have performed using our system and compare the results with other learning algorithms. We conclude in Section 5 summarizing the results and discussing future work.

2 Filtering Adaptive Boosting: MadaBoost

In this section we describe the boosting algorithm used in the learning system being evaluated in this paper.

We first justify the choice of the algorithm. The obvious choice would have been algorithm AdaBoost due to Freund and Schapire [11] since it has been repeatedly reported to outperform any other voting method [12,2,1]. AdaBoost is originally designed for the *boosting by subsampling* framework (also called boosting by re-weighting) where the base learner is required to produce a hypothesis that tries to minimize the error with respect to a weighted training set. The training set is fixed at the beginning and used throughout all the boosting steps with AdaBoost modifying the weights at each iteration depending on the hypotheses being obtained. Recall that our goal is to use a base learner that uses a small sample of the overall training set at each step so it can be run very efficiently. Thus, the boosting by subsampling framework is not appropriate for this. Instead, we should move to the so called *boosting by filtering* (also called boosting by re-sampling) where, instead of fixing a training set, we can randomly draw an example, calculate its weight and filter it according to it. In this way, we can effectively simulate the probability distribution that boosting is using at each step and obtain and un-weighted sample that has been drawn from it. This procedure is standard an is basically the sample filter that was already proposed by Freund in [10]. However, AdaBoost is not suitable for this task. If we use the weights of AdaBoost to construct a filter in the way just described, obtaining a random sample at each step with respect to the current modified distribution turns out to be very slow process. More precisely, one can show that the time taken by the filter to generate one example is exponential with respect to $\mathcal{O}(1/\epsilon_t)$, where ϵ_t is the error of the combined boosting hypothesis at the t-th step. In other words, since our goal is to reduce the error of the combined hypothesis up to a small number, this means that at some point the time taken to generate a new training set to proceed boosting one more iteration might become prohibitively large. One obvious solution is to normalize the weights so they sum up to 1 and thus, they can be used more efficiently to simulate the filtering probability (notice that in this way we will actually have the distribution under which the base learner is required to work). However, to normalize the weights we have to go through all the dataset and thus, the advantage of using sampling is simply lost. We refer the reader to [5] for a rigorous description of this problem.

Algorithm MadaBoost
Input: $\epsilon > 0$
inducer $\mathcal{ADSS}$, $T = \#\text{trials}$;
1 **for** $t = 1$ **to** T
2 $(h_t, \epsilon_t) \leftarrow \mathcal{SDS}(FilEx(t))$
3 $\beta_t \leftarrow \sqrt{\epsilon_t/(1-\epsilon_t)}$;
4 **end-for**
5 output f_T such that $f_T(x) = \text{argmax}_{y \in Y} \sum_{i:h_i(x)=y} \log \frac{1}{\beta}$
6 **Procedure** $FilEx(t)$ /* t: current boosting trial /*
7 **loop-forever**
8 generate (x, y_x) uniformly at random from training set S;
9 $\text{cons}(h_i, x) = 1$ if $h_i(x) = y_x$ and -1 otherwise;
10 $w_t(x) \leftarrow \min\{1, \prod_{i=1}^{t} \beta^{\text{cons}(h_i,x)}\}$;
11 with probability $w_t(x)$, output (x, y_x) and exit;

Fig. 1. Algorithm MadaBoost for the filtering framework.

This problem of AdaBoost has been recently addressed by Domingo and Watanabe in [5] (see also [23] where the idea was originated). For overcoming this problem, it was proposed a simple modification of AdaBoost denoted by MadaBoost that is suitable for both, the subsampling and the filtering frameworks. A description of MadaBoost for boosting by filtering together with the filter is provided in Figure 1. The filter is a loop that attempts to pass an example randomly obtained from the dataset to the base learner using the weights of the examples as a probability. The base learner algorithm $\mathcal{ADSS}$ that uses the filter is described in the following section.

In Figure 1 we can see that the only difference with the original AdaBoost algorithm is that we keep the weights always upper bounded by 1 while in AdaBoost the weights can grow unbounded. This is the key property of the modification and even in this case we can still show that the algorithm retains the original boosting property in the PAC sense. That is, if we can obtain hypotheses better than random guessing at each step, then MadaBoost can make the error of the combined hypothesis arbitrarily small in a finite amount of time. The main property of MadaBoost is that, at each boosting iteration, it can be shown that the time taken to randomly generate a new training sample under the current boosting distribution is linear in $\mathcal{O}(1/\epsilon_t)$ where ϵ_t is the error of the combined hypothesis at the t-th boosting round. Recall that this time is exponential in $\mathcal{O}(1/\epsilon_t)$ when using AdaBoost. (For more details on MadaBoost we refer the reader to [5].)

3 Adaptive Decision Stump Selector

In this section we describe the algorithm used as a base learner for the boosting process. As already discussed in the introduction, we have chosen decision stumps

as a base learner. A decision stump is a decision trees with only one internal node and two terminal nodes over discrete attributes. At the node just a single attribute is used to test whether it is equal or not to one of its possible values. Decision stumps can be also though as IF-THEN rules where the condition depends just on one attribute and one of its possible values.

Given a fixed problem description we will denote by H_{DS}, the set of all possible decision stumps over the set of attributes. To obtain any advantage through sampling we need to be able to compute set H_{DS} without looking at all the data. For this reason we consider only discrete attributes. Obviously, we also want to take into account continuous attributes, in case there is any, and for that reason we have first discretized the data. As a discretization algorithm we have used equal-width interval binning discretization with 5 intervals. Although this method has been shown to be inferior to more sophisticated methods like entropy discretization [9], it is very easy to implement, very fast and the performance difference is small [8]. We will use two different versions of the base learner, the one that does not perform sampling and uses all the data and the one that uses sampling. We will start describing the first one. Given H_{DS} and the dataset X, we will go through all the data and compute for each $h \in H_{DS}$, its error on X and output the one that has the smallest error on X. This base classifier is very similar to the 1R classifier of [13] and the MC4(1)-disc of [1] except that the discretization step is different. For the version using sampling

Algorithm $\mathcal{ADSS}$ (H_{DS}, δ);
$t \leftarrow 0$; $S \leftarrow \emptyset$; $n = |H_{DS}|$;
repeat
use $FilEx$ to generate one example and add it to S;
$t \leftarrow t + 1$;
$\alpha_t = 3\sqrt{(2\ln\tau - \ln\ln\tau + 1)/t}$, where $\tau = nt(t+1)/(2\delta\sqrt{\pi})$;
for all $h \in H_{DS}$, $U(h,S) \leftarrow \|\{x \in S : h \text{ classifies } x \text{ correctly}\}\|/t - 1/2$;
until $(\exists h \in H_{DS}$ such that $U(h,S) \geq \alpha_t)$
output $h_0 \in H_{DS}$ with largest $U(h,S)$;
output $1 - U(h_0,S)$ as an estimation of h_0's error prob.;

Fig. 2. The Adaptive Decison Stump Selector $\mathcal{ADSS}$.

we will use an algorithm proposed by Domingo et.al. in [3,4]. The algorithm is described in Figure 2 and we discuss it in the following. The algorithm, denoted by $\mathcal{ADSS}$ receives as input set H_{DS} and a confidence parameter δ. It randomly obtains examples from dataset X usign procedure $FiltEx$ described in Section 2. Every time a new example is obtained, it updates the *advantage* of every stump $h \in H_{DS}$, defined as $U(h,S) = accu_S(h) - 1/2$. Then, it decides *adaptively* when to stop sampling. The stopping condition is determined depending on: the advantages of the stumps on the current set of examples S, the number of examples t obtained so far, the number n of stumps considered and the confidence

parameter δ. The reason to consider the advantage $U(h, S)$ instead of the more commonly used accuracy $\|\{x \in S : h \text{ classifies } x \text{ correctly}\}\|/t$ is that we are going to use this algorithm as a base learner for a boosting process and boosting requires always a hypothesis that has accuracy over 50% at each iteration. While this cannot be guaranteed in general, what our sampling method guarantees is that if there exists a stump with an accuracy larger than 50%, then with high probability it will select an stump that also has an accuracy larger than 50% ensuring that the boosting process continues. The following theorem concerning the complexity and reliability of the algorithm can be easily derived from the more general results in [4] and we refer the reader to that paper for a proof.

Theorem 1. [1] *Let $\mathcal{D}$ be the filtering distribution at certain boosting step and let $h_\star \in H_{DS}$ be the hypothesis with largest advantage under that distribution, defined as $\gamma_\star = error_{\mathcal{D}}(h_\star) - 1/2$ and assume that $\gamma_\star > 0$. Then, with probability larger than $1 - \delta$, algorithm $\mathcal{ADSS}$ outputs a stump h such that $\gamma > \gamma_\star/2$, where $\gamma = error_{\mathcal{D}}(h) - 1/2$ and finishes within $\mathcal{O}((1/\gamma_\star^2)\ln(n/\delta))$*

Notice that the key property of the algorithm is that the number of examples used is not decided a priori, it depends *adaptively* on the situation. More precisely, the better the best stump at one iteration (and therefore, the larger $\gamma_\star$) the faster the algorithm will finish. Also notice that the theorem guarantees that whenever a stump with accuracy larger than 50% exists, one with accuracy also larger than 50% will be output with high probability.

We will consider two different versions of the algorithm depending on how many examples we obtain from the random sampling process at each iteration. The first version is the one provided in Figure 2, that we will refer as the *arithmetic* version, obtains just one example every iteration. The second version, the *geometric* version, increases the sample size by an s multiplicative factor at each iteration where s is a positive integer set by the user. Arithmetic and geometric sampling methods have been also studied recently in [18]. A result similar to Theorem 1 can be also proved for the geometric version where now the sample size will depend on s. We again refer the reader to [4] for more details in the theoretical properties of these algorithms.

4 Experimental Results

We have conducted our experiments on a collection of datasets from the repository at University of California at Irvine [15]. We have chosen all datasets with 2 classes since, according to previous experiments on boosting stumps, we are not expecting our base learner to be able to find weak hypothesis with accuracy better than 50% for most problems with a large number of classes. Apart from this restriction, the choice of the datasets has been done so it reflects a variety

[1] For this theorem to hold, we assume that the Central Limit Theorem applies. A slightly less efficient version but not condition to the Central Limit Theorem is also provided in [4].

of datasets sizes and combination of discrete and continuous attributes. As a test bed for comparing our boosting decision stumps algorithms we have chosen two well known learning algorithms, the decision tree inducer C4.5 [20] and the Naive Bayes classifier [7].

One point needs to be clarified on the way we used these datasets. We had to inflate the datasets since we need large datasets but most of the UCI datasets are quite small. That is, following [14], we have artificially inflated the training set (the test set has been left unchanged to avoid making the problem easier) introducing 100 copies of all records and randomizing their order. Inflating the datasets only makes the problem harder in terms of running time, does not change the problem in terms of accuracy if we restrict ourselves to use learners that are just calculating statistics about the training sample. For instance, if one particular decision stump has an accuracy of 80% on the original training set, then this stump has the same accuracy in the inflated dataset. Thus, if that is the stump of choice in the small dataset it will also be in the inflated version. Moreover, it also does not affect the results concerning sampling, neither makes the problem easier nor more difficult. The reason is that all the statistical bounds used to calculate necessary sample sizes provide results that are independent of the size of the probability space from where we are sampling that, in this case, is the training set. The necessary sample sizes depend on the probabilities on the training set and these are unchanged when we inflate the dataset. In other words, if in one particular situation the necessary sample size is 10.000, this sample size will be the same independently of whether we are sampling from the original dataset or from the inflated version.

From the above discussion we conclude the following. For the algorithms using boosting stumps and for the Naive Bayes classifier we will provide both, accuracy and running time results in the inflated datasets since those learning algorithms satisfy the requirements just described and therefore these results are meaningful. However, in the case of C4.5 we only provide running time results not accuracy results since this algorithm is not strictly based on probabilities over the sample. It makes, for instance, decisions about whether to split or not based on the actual number of instances following in one particular node and these numbers are obviously changed when we inflate the dataset. The accuracy results of C4.5 in the inflated dataset are cannot be used for comparison.

In any case, we are aware that this is perhaps not the best method to test our algorithms and real large datasets would obviously have been better; but we still believe that the results are informative and convincing enough to show the goodness of our method.

The experiments were carried out using $\delta = 0.1$ in algorithm $\mathcal{ADSS}$ and $s = 2$ for the geometric version. For every dataset, we have used a 10-fold cross validation if a test set is not provided and for the algorithms using sampling (and thus, randomized) the results are averaged over 10 runs. All the experiments have been done in a computer with a CPU alpha 600Mhz using 256Mb of memory and a hard disk of 4.3Gb running under Linux. Since enough memory was available, all the data has been loaded in a table in main memory and from there the

algorithms have been run. Loading the data took few seconds and since this time is the same for all the algorithms it has been omitted from the results. Notice that for C4.5, this in fact the only way to run it. For the other algorithms, we could have been done using the data from disk and an efficient sampling method from external memory. Doing experiments under these conditions is part of our future work. The running time results also include the time taken to construct all the decision stumps.

The experiments were done for estimating the performance of the arithmetic and geometric versions of algorithm $\mathcal{ADSS}$ combined with the version of MadaBoost for the filtering framework. For comparison purposes, we also executed MadaBoost in the boosting by subsampling framework using the whole dataset; that is, MadaBoost with a base learner that selects the best decision stump by searching through the whole dataset exhaustively. Below we use Ar., Geo., and Exh. respectively to denote the arithmetic and the geometric versions of $\mathcal{ADSS}$ and the exhaustive search selector. We have also carried out the experiments with Exh. using the original AdaBoost and found the difference in accuracy with MadaBoost with Exh. negligible (we refer the reader to [6] for a detail experimental comparison between AdaBoost and MadaBoost). We also provide the accuracy results obtained just by using a single decision stump (denoted by DS) so the gain produced by the boosting algorithm can be appreciated.

We have set the number of boosting rounds to be 10 which usually is enough to converge to a fixed training error (that is, although we keep obtaining hypothesis with accuracy slightly better than 50%, the training error does not get reduced anymore).

Table 1. Accuracies of boosted decision stumps with and withouth sampling and that of Naive Bayes.

Name	Size	$\|H_{DS}\|$	DS	Exh.	Ar.	Geo.	NB
agaricus	731100	296	88.68	97.74	97.84	98.03	98.82
kr-vs-kp	287600	222	68.24	93.19	92.89	92.71	88.05
hypothyroid	284600	192	95.70	95.86	95.84	95.86	95.43
sick-euthy.	284600	192	90.74	91.02	90.93	90.93	90.26
german	90000	222	69.90	74.10	74.28	74.30	76.00
ionos	31590	408	76.18	90.26	89.53	89.63	89.14

Table 1 shows the accuracy obtained on these datasets by three combinations of selectors with MadaBoost. The columns "Size" and "$\|H_{DS}\|$" show, for each dataset, its size and the number of all possible decision stumps respectively. As we can see easily, there is no significant difference between the accuracies obtained by these three methods. These results indicate that our sampling method is accurate enough. Two other consequences can be also observed. First, running boosting even for this few number of iterations makes a significant improvement in accuracy in most of the datasets. This can be observed by comparing the

accuracies obtained with that obtained using just a single decision stump (column DS). This accuracy has been obtained using all the dataset. Second, even though the hypotheses produced are very simple (a weighted majority of ten depth-1 decision trees), the accuracies are comparable to that obtained using Naive Bayes, a learning algorithm that has been reported to be competitive with more sophisticated methods like decision trees [7].

Table 2. Running times (in seconds) of MadaBoost with and without sampling, and that of Naive Bayes and C4.5.

Name	Exh.	Ar.	Geo.	NB	C4.5
agaricus	892.31	2.07	1.78	16.34	21.65
kr-vs-kp	265.63	3.68	2.75	10.07	31.13
hypothyroid	233.24	5.82	5.67	7.14	67.40
sick-euthy.	232.05	6.84	6.77	7.08	162.76
german	80.75	16.96	10.47	1.08	20.34
ionos	56.95	6.29	4.74	0.85	29.47

Once we have established that there is no loss in accuracy due to the use of sampling, we should check whether there is any gain in efficiency. Table 2 shows the running times of MadaBoost combined with three selection algorithms, exhaustive one (Exh.), and the arithmetic (Ar.) and the geometric (Geo.) versions. We have also provided the running time of Naive Bayes (NB) and C4.5 for those datasets.

Let us discuss about these results. First, one can easily see that the exhaustive search version is a very slow process, particularly for large datasets. The running time of MadaBoost with Exh. is a function of the dataset size, the number of decision stumps considered (which depends on how many attributes the dataset has and their range of values), and the number of boosting rounds.

For the algorithms using sampling, we can see that the running time has been greatly reduced for all datasets. The running time of MadaBoost with the arithmetic version of $\mathcal{ADSS}$ is, on average, approximately 40 times smaller than that of the Exh. Surprisingly enough, for the sampling versions the fastest dataset becomes the largest one, Agaricus. The reason is the particular structure of this dataset. During all the 10 boosting iterations one can find hypothesis with accuracy larger than 70% and thus, the sample sizes needed at each step are very small. This contrasts with datasets like German where a similar number of decision stumps is considered and, even though the dataset is less than 1/8 of Agaricus, the running time on German is 8 times larger. This is because for this dataset, after the third boosting iteration, even the best stump has accuracy smaller than 60% and this affects the efficiency of the sampling method. Recall that the inverse of the advantage of the best stump (i.e. the distance of its accuracy compare to the random hypothesis) is the major term that determines the necessary sample size at each boosting round.

Table 3. Running time of $\mathcal{ADSS}$ with geometric sampling for different values of s.

Name	$s = 1.5$	$s = 2$	$s = 2.5$	$s = 3$
agaricus	1.47	1.78	2.01	2.11
kr-vs-kp	2.33	2.75	3.07	2.93
hypothyroid	5.16	5.67	6.67	7.04
sick-euthy.	6.48	6.77	6.79	7.01
german	9.05	10.47	10.41	13.17
ionos	4.27	4.74	5.29	6.06

This difference between Agaricus and German becomes more clear by looking at the total number of examples used by a base learner, i.e., $\mathcal{ADSS}$, during the boosting iterations. Recall that we are using boosting by filtering; so when the base learner asks for an example, the filtering method might have to actually sample several of them until it gets one that passes the filter and is given to the base learner. Our experiment shows that, in total, German needs 264,763 examples while Agaricus needs only 82,915 examples. On the other hand, the number of examples actually used by the base learner is 162,058 for German and 13,907 for Agaricus, and there is more than 10 times difference.

As for the difference between arithmetic and geometric sampling, we can see that, for this particular problem and our choice of $s = 2$, the geometric sampling is, on average, around 1.3 times faster than the arithmetic sampling, which is what we can expect from our theoretical estimation provided in [4]. In Table 3 we provide the running times of Geo. for other values of s.

Table 4. Accuracies and running times for 10, 20 and 30 boosting rounds.

DataSet	Running Time			Accuracy		
Name	10	20	30	10	20	30
agaricus	1.78	5.97	13.88	98.03	98.95	99.77
kr-vs-kp	2.75	8.44	20.88	92.71	93.48	93.94
hypothyroid	5.38	33.31	128.29	95.86	95.87	95.86
sick-euthy.	6.77	58.17	172.56	90.93	91.08	92.13
german	10.47	20.34	41.21	74.30	76.37	77.10
ionos	4.74	13.98	28.29	89.63	92.35	93.90

With respect to C4.5, we can see that our algorithm is faster in many datasets. More specifically, MadaBoost with $\mathcal{ADSS}$ using geometric sampling is around 10 times faster than C4.5. Compared to Naive Bayes, we can see that our method is much faster for the largest datasets (more than 9 times for Agaricus) but as the dataset becomes smaller the advantage is lost and, for instance, for dataset German our method becomes around 10 times slower. Notice that Naive Bayes is perhaps the fastest known learning method that uses all the dataset since its running time scales exactly linear in the training set.

We have also done some experiments to check the scalability of the process with respect to the number of boosting iterations. The results are shown in Table 4. These results have been obtained using MadaBoost and the geometric version of Decision Stumps Selector with $s = 2$. Notice that for some datasets there is still a gain in accuracy and that, in general, is still much faster to run 30 boosting iterations using our sampling method than to run 10 boosting iterations using all the dataset.

5 Conclusions

In this paper we have studied a simple learning system based on boosting decision stumps. Our goal was to show that this learning system can be executed efficiently using a large dataset. For achieving this goal we have proposed the following two tools. First, we have use an adaptive sampling method for selecting the decision stump at each iteration so that this process can be done using only part of the dataset. Second, we have use a modification of AdaBoost that can be efficiently run under the boosting by filtering method and thus, appropiately combined with the sampling based base learner that we used. We have provide experimental evidence that this method is much more efficient than using all the dataset and that there is no apparent lost in accuracy.

Future work involves finding a stronger based learner that, while we can still combine with our sampling method provides us with a better accuracy at each step so that the overall accuracy is better and we can also use it for problems with a large number of classes. Another interesting topic is to perform these experiments but using much larger datasets that cannot be loaded in main memory and thus, we have to sample from external memory.

References

1. Bauer, E. and Kohavi, R. 1998. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 1–38, 1998.
2. Dietterich, T., 1998. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 32:1–22.
3. Domingo, C., Gavaldà, R. and Watanabe, R., 1998. Practical Algorithms for On-line Selection. *Proceedings of the First International Conference on Discovery Science*, DS'98. Lecture Notes in Artificial Intelligence 1532:150–161.
4. Domingo, C., Gavaldà, R. and Watanabe, O., 1999. Adaptive Sampling Methods for Scaling Up Knowledge Discovery Algorithms. *Proceedings of the Second International Conference on Discovery Science*, DS'99. Lecture Notes on Artificial Intelligence, 1721, pp. 172-183.
5. Domingo, C. and Watanabe, O., 1999. MadaBoost: A modification of AdaBoost. Tech Rep. C-133, Dept. of Math and Computing Science, Tokyo Institute of Technology. URL: `http://www.is.titech.ac.jp/~carlos`.

6. Domingo, C. and Watanabe, O., 1999. Experimental evaluation of a modification of AdaBoost for the filtering framework. Tech Rep. C-139, Dept. of Math and Computing Science, Tokyo Institute of Technology. URL: `http://www.is.titech.ac.jp/~carlos`.
7. P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Machine Learning*, 29:2, 103–130, 1997.
8. Dougherty, J., Kohavi, R., and Sahami, M., 1995. Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the Twelfth International Conference on Machine Learning.*
9. Fayad, U.M. and Irani, K.B., 1993. Multi-interval discretization of continuous-valued attributes for classification learning. *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 1022-1027.
10. Freund, Y., 1995. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285.
11. Freund, Y., and Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS*, 55(1):119–139.
12. Freund, Y., and Schapire, R.E., 1997. Experiments with a new boosting algorithm. *Proceedings of the 13th International Conference on Machine Learning*, 148–146.
13. R.C. Holte. Very simple classification rules perform well on most common datasets. *Machine Learning*, 11:63–91, 1993.
14. John, G. H. and Langley, P., 1996. Static Versus Dynamic Sampling for Data Mining. *Proc. of the Second International Conference on Knowledge Discovery and Data Mining*, AAAI/MIT Press.
15. Keogh, E., Blake, C. and Merz, C.J., 1998. UCI repository of machine learning databases, [http://www.ics.uci.ed u/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.
16. Lipton, R. J. and Naughton, J. F., 1995. Query Size Estimation by Adaptive Sampling. *Journal of Computer and System Science*, 51:18–25.
17. Lipton, R. J., Naughton, J. F., Schneider, D. A. and Seshadri, S., 1995. Efficient sampling strategies for relational database operations. *Theoretical Computer Science*, 116:195–226.
18. Provost, F., Jensen, D. and Oates, T., 1999. Efficient Progressive Sampling. *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining.*
19. Quinlan, J. R., 1996. Bagging, Boosting and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, AAAI Press and the MIT Press, pp. 725–723.
20. Quinlan, J. R., 1993. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Mateo, California.
21. Schapire, R. E., 1990. The strength of weak learnability. *Machine Learning*, 5(2):197–227.
22. Wald, A., 1947. *Sequential Analysis.* Wiley Mathematical, Statistics Series.
23. Watanabe, O., 1999. From Computational Learning Theory to Discovery Science. *Proc. of the 26th International Colloquim on Automata, Languages and Programming*, ICALP'99 Invited talk. Lecture Notes in Computer Science 1644:134–148.

Adaptive Boosting for Spatial Functions with Unstable Driving Attributes*

Aleksandar Lazarevic[1], Tim Fiez[2], and Zoran Obradovic[1]

[1] School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, USA

{alazarev, zoran}@eecs.wsu.edu

[2] Department of Crop and Soil Sciences, Washington State University, Pullman, WA 99164-2752, USA

tfiez@wsu.edu

Abstract. Combining multiple global models (e.g. back-propagation based neural networks) is an effective technique for improving classification accuracy by reducing a variance through manipulating training data distributions. Standard combining methods do not improve local classifiers (e.g. k-nearest neighbors) due to their low sensitivity to data perturbation. Here, we propose an adaptive attribute boosting technique to coalesce multiple local classifiers each using different relevant attribute information. In addition, a modification of boosting method is developed for heterogeneous spatial databases with unstable driving attributes by drawing spatial blocks of data at each boosting round. To reduce the computational costs of k-nearest neighbor (k-NN) classifiers, a novel fast k-NN algorithm is designed. The adaptive attribute boosting applied to real life spatial data and artificial spatial data show observable improvements in prediction accuracy for both local and global classifiers when unstable driving attributes are present in the data. The "spatial" variant of boosting applied to the same data sets resulted in highly significant improvements for the k-NN classifier, making it competitive to boosted neural networks.

Keywords: multi-strategy learning, boosting, attribute representation, spatial databases, fast k-NN classifier.

1. Introduction

Many large-scale data analysis problems involve an investigation of relationships between attributes in heterogeneous databases. Large data sets very often exhibit attribute instability, such that the set of relevant attributes is not the same through the entire data space. This is especially true in spatial databases, where different spatial regions may have completely different characteristics.

It is known in machine learning theory that combining multiple classifiers is an effective technique for improving prediction accuracy. There are many general

* Partial support by the INEEL University Research Consortium project No. C94-175936 to T. Fiez and Z. Obradovic is gratefully acknowledged.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 329-340, 2000.
© Springer-Verlag Berlin Heidelberg 2000

combining algorithms such as bagging [1], boosting [2], or Error Correcting Output Codes (ECOC) [3] that significantly improve global classifiers like decision trees, rule learners, and neural networks. These algorithms may manipulate the training patterns individual classifiers use (bagging, boosting) or the class labels (ECOC). An ensemble of classifiers must be both diverse and accurate in order to improve accuracy of the whole. Diversity is required to ensure that all the classifiers do not make the same errors. In order to increase the diversity of combined classifiers for spatial heterogeneous databases with attribute instability, one cannot assume that the same set of attributes is appropriate for each single classifier. For each training sample, drawn in a bagging or boosting iteration, a different set of attributes is relevant and therefore the appropriate attribute set should be used by each single classifier in an iteration. In addition, the application of different classifiers on spatial databases, where the data are highly spatially correlated, may produce spatially correlated errors. In such situations the standard combining methods might require different schemes for manipulating the training instances in order to keep the diversity of classifiers.

In this paper, we propose a modification of the AdaBoost algorithm [2] for combining multiple classifiers to improve overall classification accuracy. In each boosting round we try to maximize the local information for a drawn sample by changing attribute representation through attribute selection, attribute extraction and appropriate attribute weighting methods [4]. In order to exploit the spatial data knowledge, a modification of the boosting method appropriate for heterogeneous spatial databases is proposed, where at each boosting round spatial data blocks are drawn instead of the standard approach of sampling single instances.

The influence of these adjustments to single classifiers is not the same for local classifiers (e.g. k–nearest neighbor) and global classifiers (e.g. artificial neural networks). It is known that standard combining methods do not improve simple local classifiers due to correlated predictions across the outputs from multiple combined classifiers [1, 3]. We show that prediction of combined nearest neighbor classifiers can be decorrelated by selecting different attribute representations for each sample and by sampling spatial data blocks. The nearest neighbor classifier is often criticized for slow run-time performance and large memory requirements, and using multiple nearest neighbor classifiers could further worsen the problem. Therefore, we used a novel fast method for k-nearest neighbor classification to speed up the boosting process. We also test the influence of changing attribute representation on global classifiers like neural networks.

2. Related Work

The nearest neighbor classifier [6] is one of the oldest and simplest methods for performing general, non-parametric classification. A common extension is to choose the most common class among the k nearest neighbors. Despite its simplicity, the k-nearest neighbor classifier (k-NN) can often provide similar accuracy to more sophisticated methods such as decision trees or neural networks. It's advantages include ability to learn from a small set of examples, and to incrementally add new information at runtime.

Recently, researchers have begun testing methods for improving classification accuracy by combining multiple versions of a single classifier, also known as an ensemble approach. Unfortunately, many combining methods do not improve the k-NN classifier. For example, when experimenting with bagging, Breiman [1] found no difference in accuracy between the bagged k-NN classifier and the single model approach. It is also shown that ECOC will not improve classifiers that use local information due to high error correlation [3].

A popular alternative to bagging is boosting, which uses adaptive sampling of patterns to generate the ensemble. In boosting [2], the classifiers in the ensemble are trained serially, with the weights on the training instances set adaptively according to the performance of the previous classifiers. The main idea is that the classification algorithm should concentrate on the difficult instances. Boosting can generate more diverse ensembles than bagging does, due to its ability to manipulate the input distributions. However, it is not clear how one should apply boosting to the k-NN classifier for the following reasons: (1) boosting stops when a classifier obtains 100% accuracy on the training set, but this is always true for the k-NN classifier, (2) increasing the weight on a hard to classify instance does not help to correctly classify that instance as each prototype can only help classify its neighbors, not itself. Freund and Schapire [2] applied a modified version of boosting to the k-NN classifier that worked around these problems by limiting each classifier to a small number of prototypes. However, their goal was not to improve accuracy, but to improve speed while maintaining current performance levels.

Although there is a large body of research on multiple model methods for classification, very little specifically deals with combining k-NN classifiers. Ricci and Aha [5] applied ECOC to the k-NN classifier (NN-ECOC). Normally, applying ECOC to k-NN would not work since the errors in two-class problems would be perfectly correlated. However, they found that applying attribute selection to the two-class problems decorrelated errors if different attributes were selected. Unlike this approach, Bay's Multiple Feature Subsets (MFS) method [6] uses random attributes when combining individual classifiers by simple voting. Each time a pattern is presented for classification, a new random subset of attributes is selected for each classifier.

Although it is known that boosting works well with global classifiers like neural networks, there have been several experiments in selecting different attribute subsets as an attempt to force the classifiers to make different and hopefully uncorrelated errors. Tumer and Ghosh [7] found that with neural networks, selectively removing attributes could decorrelate errors. Unfortunately, the error rates in the individual classifiers increased, and as a result there was little or no improvement in the ensemble. Cherkauer [8] was more successful, and was able to combine neural networks that used different hand selected attributes to achieve human expert level performance in identifying volcanoes from images.

3. Methodology

3.1 Adaptive Boosting for k-NN Classifiers

We follow the generalized procedure of AdaBoost.M2 [2]. The modified algorithm is shown in Fig. 1. It maintains a distribution D_t over the training examples, which can be initially uniform. The algorithm proceeds in a series of T rounds. In each round, the entire weighted training set is given to the weak learner to compute weak hypothesis h_t. The distribution is updated to give wrong classifications higher weights than correct classifications.

- Given: Set S $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, k\}$
- Initialize the distribution D_t over the examples, such that $D_1(i) = 1/m$.
- For t = 1, 2, 3, 4, ... T
  0. Find relevant feature information for distribution D_t
  1. Train weak learner using distribution D_t
  2. Compute weak hypothesis h_t: $X \times Y \rightarrow [0, 1]$
  3. Compute the pseudo-loss of hypothesis h_t:

$$\varepsilon_t = \frac{1}{2} \cdot \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$$

  4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$
  5. Update D_t: $D_{t+1}(i, y) = (D_t(i, y) / Z_t) \cdot \beta_t^{(1/2)\cdot(1 - h_t(x_i, y_i) + h_t(x_i, y_i))}$
     where Z_t is a normalization constant chosen such that D_{t+1} is a distribution.
- Output the final hypothesis: $h_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$

Fig. 1. The scheme of modified AdaBoost.M2 algorithm

Since at each boosting iteration t we have different training samples drawn according to the distribution D_t, at the beginning of the "for loop" in Fig. 1 we modify the standard algorithm by adding **step 0.**, wherein we choose a different attribute representation for each sample. Different attribute representations are realized through attribute selection, attribute extraction and attribute weighting processes through boosting iterations. This is an attempt to force individual classifiers to make different and hopefully uncorrelated errors.

Error correlation is related to Breiman's [1] concept of stability in classifiers. Nearest neighbor classifiers are stable to the patterns, so bagging and boosting generate poor k-NN ensembles. Nearest neighbor classifiers, however, are extremely sensitive to the attributes used. Our approach attempts to use this instability to generate a diverse set of local classifiers with uncorrelated errors. At each boosting round, we perform one of the following methods to determine a suitable attribute space for use in classification.

To eliminate irrelevant and highly correlated attributes, regression-based attribute selection was performed through performance feedback [4] forward selection and backward elimination search techniques based on linear regression mean square error (MSE) minimization. The r most relevant attributes are selected according to the selection criterion at each round of boosting, and are used by the k-NN classifiers.

In contrast to attribute selection where a decision is target-based, variance-based dimensionality reduction through attribute extraction is also considered. Here, linear Principal Components Analysis (PCA) [4] was employed. Each of the k-NN classifiers uses the same number of new transformed attributes. Another possibility is to choose an appropriate number of newly transformed attributes which will retain some predefined part of the variance.

The attribute weighting method used in the proposed method is based on a 1-layer feedforward neural network. First, we try to perform target value prediction for the drawn sample with defined a 1-layer feedforward neural network using all attributes. It turns out that this kind of neural network can discriminate relevant from irrelevant attributes. Therefore, the neural networks interconnection weights are taken as attribute weights for the k-NN classifier.

To further experiment with attribute stability properties, miscellaneous attribute selection algorithms [4] were applied on the entire training set and the most stable attributes were selected. Then the standard boosting method was applied to the k-NN classifiers using the identified fixed set of attributes at each boosting iteration. When boosting is applied with attribute selection at each boosting round, the attribute occurrence frequency is monitored in order to compare the most stable selected attributes. When attribute subsets selected through boosting iterations become stable, this can be an indication to stop the boosting process.

3.2 Spatial Boosting for k-NN Classifiers

Spatial data represent a collection of attributes whose dependence is strongly related to a spatial location where observations close to each other are more likely to be similar than observations widely separated in space. Explanatory attributes, as well as the target attribute in spatial data sets are very often highly spatially correlated. As a consequence, applying different classification techniques on such data is likely to produce errors that are also spatially correlated [12]. Therefore, when applied to spatial data, the boosting method may require different partitioning schemes than simple weighted selection which doesn't take into account the spatial properties of the data. Rather than drawing n data points according to the distribution D_t (Fig. 1), the proposed method draws (n/M^2) spatial data blocks (squares of size M points × M points) according to the new distribution SD_t. The distribution SD_t is modified in such a way that all data points dp inside the same spatial block have the same $SD_t(dp)$. This is done through simple median M × M filtering over the data points inside the spatial block. Using this approach we hope to achieve more decorrelated classifiers whose integration can further improve model generalization capabilities for spatial data.

3.3 Adaptive Attribute and Spatial Boosting for Neural Network Classifiers

Although standard boosting can increase prediction accuracy of artificial neural network classifiers, we experimented with changing attribute representation and spatial block drawing to see if adaptive attribute and spatial boosting can further improve accuracy of an ensemble of classifiers. The most stable attributes used in standard boosting of k-NN classifiers were also used here for the same purpose. At each boosting round we performed attribute selection and attribute extraction processes, since the attribute weighting method seemed to be "redundant" when training neural network classifiers. We trained multilayer (2-layered) feedforward neural network classification models with the number of hidden neurons equal to the number of input attributes. We also experimented with different numbers of hidden neurons. The neural network classification models had the number of output nodes equal to the number of classes (3 in our experiments), where the predicted class is from the output with largest response. We used two learning algorithms: resilient propagation [10] and Levenberg-Marquardt [11]. The experiments for testing attribute stability through the boosting were repeated as well, and they were used to determine the proper number of boosting iterations.

3.4 The Fast k-NN Algorithm

The quality of k-NN generalization depends on which k instances are deemed least distant, which is determined by its distance function. We consider two distance functions in our work: standard Euclidean and Mahalanobis distance.

To speed up the long-lasting boosting process, a fast k-NN classifier is proposed. For n training examples and d attributes our approach requires preprocessing which takes $O(d \cdot n \cdot log\ n)$ steps to sort each attribute separately. However, this is performed only once, and we trade off this initial time for later speedups.

The main idea of the proposed algorithm will be presented is as follows.

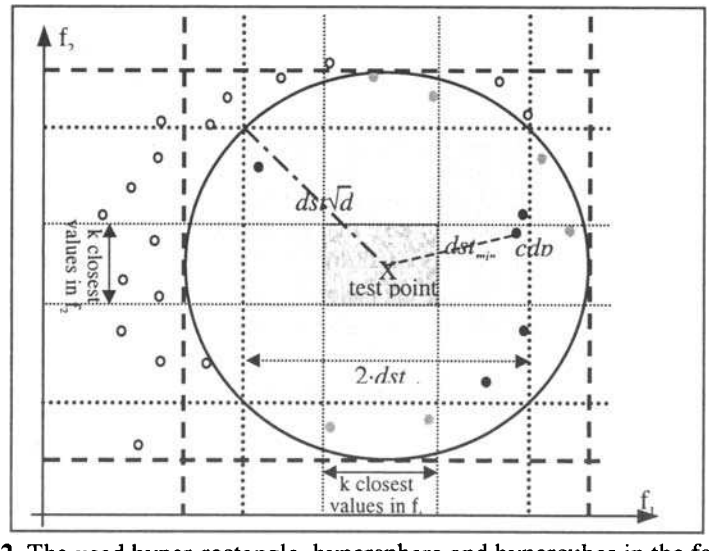


Fig. 2. The used hyper-rectangle, hypersphere and hypercubes in the fast k-NN

Initially, we form a hyper-rectangle with boundaries defined by the extreme values of the k closest values for each attribute (Fig. 2 – small dotted lines). If the number of training instances inside the identified hyper-rectangle is less than k, we compute the distances from the test point to all of $d \cdot k$ data points which correspond to the k closest values for each of d attributes, and sort them into non-decreasing array sx. We take the nearest training example cdp with the distance dst_{min}, and form a hypercube with boundaries defined by this minimum distance dst_{min} (Fig. 2 - larger dotted lines). If the hypercube doesn't contain enough (k) training points, form the hypercube of a side $2 \cdot sx(k+1)$. Although this hypercube contains more than k training examples, we need to find the one which contains the minimal number of training examples greater than k. Therefore, if needed, we search for a minimal hypercube by binary halving the index in non-decreasing array sx. This can be executed at most $log\ k$ times, since we are reducing the size of the hypercube from $2 \cdot sx(k+1)$ to $2 \cdot sx(1)$. Therefore the total time complexity of our algorithm is $O(d \cdot log\ k \cdot log\ n)$, under the assumption that $n > d \cdot k$, which is always true in practical problems.

If the number of training instances inside the identified hyper-rectangle is greater than k, we also search for a minimal hypercube that contains at least k and at most $2 \cdot k$ training instances inside the hypercube. This was accomplished through binary halving or incrementing the side of a hypercube. After each modification of a hypercube's side, we compute the number of enclosed training instances and modify the hypercube accordingly. By analogous reasoning as in the previous case, it can be shown that binary halving or incrementing the hypercube's side will not take more than $log\ k$ time, and therefore the total time complexity is still $O(d \cdot log\ k \cdot log\ n)$.

When we find a hypercube which contains the appropriate number of points, it is not necessary that all k nearest neighbors are in the hypercube, since some of the closer training instances to the test points could be located in a hypersphere of identified radius $dst\sqrt{d}$ (Fig. 2). Since there is no fast way to compute the number of instances inside the sphere without computing all the distances, we embed the hypersphere in a minimal hypercube and compute the number of the training points inside this surrounding hypercube. The number of points inside the surrounding hypercube is much less than the total number of training instances and therefore classification speedups of our algorithm.

4. Experimental Results

Our experiments were performed using spatial data from a 220 ha field located near Pullman, WA. All attributes were interpolated to a 10x10 m grid resulting in 24,598 patterns. The Pullman data set contained x and y coordinates, 19 soil and topographic attributes and the corresponding crop yield. We also performed the experiments on an artificial data set made using our spatial data simulator [13] corresponding to 5 heterogeneous data distributions, each having different relevant attributes for yield generation. The data set had 5 relevant and 5 irrelevant attributes.

For the Pullman data set the miscellaneous attribute selection methods were used to identify the 4 most relevant attributes (Table 1) and the most stable attributes (4, 7, 9, 20) were used for the standard boosting method.

Table 1. Attribute selection methods used to identify 4 most stable attributes

Attribute Selection Methods						
Forward Selection			*Selected Attributes*	*Branch and Bound*		*Selected Attributes*
Inter-class distance	Minkowski order	1	7, 9, 10, 11	Probabilistic distance	Bhatacharya	4, 7, 10, 14
		3	3, 4, 5, 7		Mahalanobis	7, 9, 11, 20
	Euclidean		3, 4, 5, 7		Patrick-Fischer	13,17,20,21
	Chebychev		3, 4, 5, 7			
Probabilistic distance	Bhatacharya		3, 4, 8, 9	*Backward Elimination*		*Selected Attributes*
	Mahalanobis		7, 9, 11, 20			
	Divergence metric		3, 4, 8, 9	Probabilistic distance	Bhatacharya	4, 7, 9, 14
	Patrick-Fischer		13,16,20,21		Mahalanobis	7, 9, 11, 20
Minimal Error Probability, k-NN with Resubstitution			4, 7, 11, 19		Patrick-Fischer	13,17,20,21
Linear Regression Performance Feedback			5, 9, 7, 18	Linear Regression Performance Feedback		7, 9, 11, 20

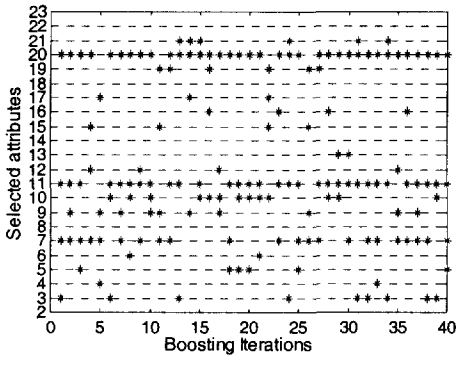


Fig. 3. Attribute stability during boosting on k-NN classifiers

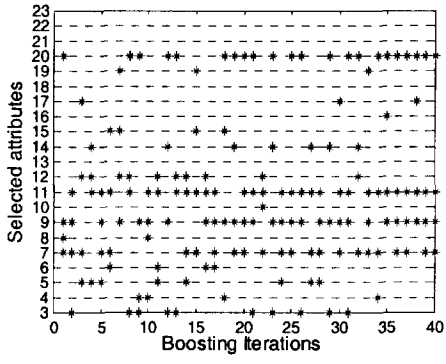


Fig. 4. Attribute stability during boosting on Levenberq-Marquardt algorithm

For the k-NN classifier experiments, the value of k was set using cross validation performance estimates on the entire training set. The selected attributes during the boosting iterations were monitored and their frequency was computed. The attribute frequency during the boosting rounds for backward elimination is shown in Fig. 3.

PCA was also used to reduce data dimensionality at each boosting round. Here, projections to 4 dimensions explained most of the variance and there was little improvement from additional dimensions. For the attribute weighting method, we used the attribute weights given by a neural network. For each of these attribute representation changing methods, boosting was applied to k-NN classifiers and the classification accuracies for 3 equal size classes are given in Table 2.

Table 2. Comparative analysis of overall classification test accuracies for 3-class problems

Number of Boosting Rounds	Standard Boosting on k-NN	Attribute Boosting on k-NN with				Boosting on	
		Forward Selection	Backward Elimination	PCA	Attribute Weighting	Levenberg-Marquardt	
						Standard Boosting	Backward Elimination
8	38.2	40.9	38.5	42.4	43.0	43.6	47.5
16	39.5	41.3	38.8	42.4	43.9	44.1	47.8
24	38.8	41.9	42.1	44.5	44.8	44.8	48.3
32	38.5	41.8	43.5	45.1	46.1	45.5	48.8
40	39.3	42.1	42.8	43.4	44.3	44.9	48.5

Analyzing the data from Table 2, the methods of adaptive attribute boosting outperformed the standard boosting model. The results indicate that 30 boosting rounds were usually sufficient to maximize prediction accuracy. After this many iterations, attribute selection somewhat stabilized although attribute selection during boosting was less stable for k-NN (Fig. 3) than for neural networks (Fig. 4). For k-NN after approximately 30 boosting rounds the attributes became fairly stable with attributes 7, 11 and 20 obviously more stable than attributes 3 and 9 which also appeared in later iterations. The prediction accuracies for k-NN classifier experiments using Mahalanobis distance were worse than those using k-NN classifier with Euclidean distance, and are not reported here. The results show that our approach is promising. For each method of changing attribute representation, we achieve better prediction accuracy than using the standard boosting method.

The frequency of selected attributes during the boosting rounds when boosting was applied to neural network classification models is presented in Fig. 4. The best results were obtained with applied backward elimination attribute selection using the Levenberq-Marquardt algorithm (Table 2). It appears that monitoring selected attributes can be a good criterion for early stopping of boosting, since after the selected attribute subsets become stable no significant improvements in prediction accuracy was noticed. In this case it was even more evident that attributes stabilized after approximately 30 boosting rounds. During the boosting iterations we were selecting the 4 and 5 most important attributes, and the number of hidden neurons in a 2-layer feedforward neural network was equal to the number of input attributes. We noticed that further increasing the number of hidden neurons did not improve prediction accuracy probably because of overfitting.

Since our data have a spatial dimension, we also performed experiments with a modified spatial boosting method. Applying the spatial boosting method to a k-NN classifier, we achieved much better prediction than using the previous two boosting methods on a k-NN classifier (Table 3). Furthermore, when applying spatial boosting with attribute selection at each round, the prediction accuracy was increased slightly as the size (M) of the spatial block was increased (Table 3). No such improvements were noticed for spatial boosting with fixed attributes or with the attribute weighting method, and therefore the classification accuracies for just M = 5 are given.

Applying spatial boosting on neural network classifiers resulted in no enhancements in classification accuracies. Moreover, for pure spatial boosting without attribute selection we obtained slightly worse classification accuracies than using "non-spatial" boosting. This phenomenon is due to spatial correlation of our attributes, which means that data points close in the attribute space are probably close in real space. Since k-NN examines this local information, it gains from spatial data blocks unlike neural networks which do not consider any kind of spatial information during the training. Therefore, one of our current concerns will be to find a technique to include spatial knowledge into the training of neural networks classifiers.

Table 3. Overall accuracy of spatial boosting on a 3-class real-life test data using k-NN

Number of Boosting Rounds	Spatial Boosting for k-NN with					
	Fixed Attribute Set	Backward Elimination Attribute Selection				Attribute Weighting
	M = 5	M = 2	M = 3	M = 4	M = 5	M = 5
8	46.4	45.8	47.7	48.1	47.8	45.2
16	46.6	46.2	47.6	48.1	47.7	45.6
24	46.7	46.7	47.9	48.2	48.2	45.8
32	46.9	46.9	48.1	48.4	47.9	46.3
40	47.0	47.2	48.1	47.9	47.8	45.9

Although we achieved promising results on the real life data, we repeated all experiments for the more controllable artificial data set, which had 5 clusters similar in attribute space. Each of these clusters had a different set of relevant attributes used for yield generation. The best results for boosting of k-NN and neural network classifiers are shown in Table 4.

The adaptive attribute boosting results show no improvements in prediction accuracy, which was due to properties of the artificial data set. Each different region has not only different relevant attributes related to yield class but also a different number of relevant attributes. Since we are not sure of the number of relevant attributes for each region, we need to select at least the 4 or 5 most important attributes at each boosting round. However, the total number of relevant attributes in the data set is 5 as well, and therefore we could not achieve any attribute instability. To avoid forcing the standard boosting method to be inferior to our method, we used all 5 relevant attributes from the data set for standard boosting. If we select the 5 best

attributes during each boosting iteration, it is obvious that we will achieve similar results. Therefore, we were selecting the 4 most relevant attributes knowing that for some drawn samples we would lose beneficial information. Nevertheless, we obtained similar classification accuracies as the standard boosting method, but reached the "bounded" final prediction accuracy in a smaller number of boosting iterations. This could be very important in order to reduce the time needed for the latest boosting rounds. Instead of post-pruning the boosted classifiers [14] we can try to on-line settle the appropriate number of boosting iterations.

Table 4. Comparative analysis of overall classification accuracies for 3-class problems on artificial test data set with 5 clusters (BE stands for Backward Elimination, LM for Levenberg-Marquardt algorithm)

Number of Boosting Rounds	Boosting Applied to k-NN Classifier		Boosting Applied to LM Neural Networks		Spatial Boosting for k-NN with				
					Fixed Attribute Set	Backward Elimination Attribute Selection			
	Standard	BE	Standard	BE	M = 5	M=2	M=3	M=4	M=5
8	57.9	57.5	65.3	66.1	65.6	64.6	65.3	65.4	66.0
16	59.1	59.1	66.7	67.2	65.5	65.2	65.9	65.2	66.7
24	57.6	58.7	67.1	69.3	65.8	65.5	65.9	65.8	67.0
32	58.3	58.5	68.8	69.2	66.0	65.4	66.2	66.1	67.6
40	58.2	59.2	69.8	69.4	66.1	65.3	66.4	66.7	68.1

Classification accuracies of spatial boosting for k-NN on the artificial data set were again much better than without using spatial information and comparable to neural networks. Here, the classification accuracy improvements from increasing the size (M) of the spatial blocks were more apparent than for real-life data due to the higher spatial correlation of the artificial data.

5. Conclusions and Future Work

Results from two spatial data sets indicate that the proposed algorithm for combining multiple classifiers can result in significantly better predictions over existing classifier ensembles, especially for heterogeneous data sets with attribute instabilities. By manipulating the attribute representation used by individual classifiers at each boosting round, we showed that classifiers could be more decorrelated thus leading to higher prediction accuracy. The attribute stability test served as a good indicator for proper stopping of further boosting iterations. Testing of the proposed method seems to indicate that a smaller number of iterations is needed in order to achieve the same final prediction accuracy. The new boosting method proposed for spatial data showed promising results for k-NN classifiers making it competitive with highly non-linear and powerful models like neural networks.

In this paper, we concentrated on improving the accuracy of the global classifier. Although the new fast k-NN classifier significantly reduces the computational requirements, an open research question is to further increase the speed of ensembles of k-NN classifiers for high-dimensional data.

Although the performed experiments provide evidence that the proposed approach can improve predictions of ensemble of classifiers, further work is needed to examine the method for more heterogeneous data sets with more diverse attributes. In addition, we are working to extend the method to regression based problems.

References

1. Breiman, L.: Bagging predictors, Machine Learning 24, 123-140, (1996)
2. Freund, Y., and Schapire, R. E.: Experiments with a new boosting algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, pp. 325-332, (1996)
3. Kong, E. B., Dietterich, T. G.: Error-correcting output coding corrects bias and variance, In Proc. of the twelfth National Conference on Artificial Intelligence, 725-730, (1996)
4. Liu, L. and Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, Boston (1998)
5. Ricci, F., and Aha, D. W.: Error-correcting output codes for local learners, In Proceedings of the 10th European Conference on Machine Learning, (1998)
6. Bay, S. D.: Nearest Neighbor Classification from Multiple Feature Subsets. Intelligent Data Analysis. 3(3):191-209, (1999)
7. Tumer, K., and Ghosh, J.: Error correlation and error reduction in ensemble classifiers, Connection Science 8, 385-404, (1996)
8. Cherkauer, K. J.: Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In P. Chan, (Ed.): Working Notes of the AAAI Workshop on Integrating Multiple Learned Models, 15-21, (1996)
9. Bishop, C., Neural Networks for Pattern Recognition, Oxford University Press, (1995)
10. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, Proceedings of the IEEE International Conf. on Neural Networks, San Francisco, 586–591 (1993)
11. Hagan, M., Menhaj, M.B.: Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks 5, 989-993 (1994)
12. Vucetic, S., Fiez, T., Obradovic, Z.: A Data Partitioning Scheme for Spatial Regression, Proceedings of the IEEE/INNS Int'l Conf. on Neural Networks, Washington, D.C., No. 348, session 8.1A., (1999)
13. Pokrajac, D., Fiez, T. and Obradovic, Z.: A Spatial Data Simulator for Agriculture Knowledge Discovery Applications, in review.
14. Margineantu, D. D., and Dietterich, T. G.: Pruning adaptive boosting, In Proceedings of the 14th International Conference on Machine Learning, 211-218 (1997)

Robust Ensemble Learning for Data Mining*

Gunnar Rätsch[1], Bernhard Schölkopf[2], Alexander Johannes Smola[3], Sebastian Mika[1], Takashi Onoda[4], and Klaus-Robert Müller[1]

[1] GMD FIRST, Kekuléstr. 7, D-12489 Berlin, Germany
[2] Microsoft Research, 1 Guildhall Street, Cambridge, UK
[3] Dep. of Engineering, ANU, Canberra ACT 0200, Australia
[4] CIRL CRIEPI, 2-11-1 Iwadokita, Komae-shi, Tokyo, 201-8511 Japan
{raetsch,mika,klaus}@first.gmd.de, bsc@microsoft.com,
Alex.Smola@anu.edu.au, onoda@criepi.denken.or.jp

Abstract. We propose a new boosting algorithm which similarly to ν-Support-Vector Classification allows for the possibility of a pre-specified fraction ν of points to lie in the margin area or even on the wrong side of the decision boundary. It gives a nicely interpretable way of controlling the trade-off between minimizing training error and capacity. Furthermore, it can act as a filter for finding and selecting informative patterns from a database.

1 Introduction

Boosting and related Ensemble learning methods have been recently used with great success in applications such as Optical Character Recognition [2, 3, 11]. The idea of a large (minimum) margin explains the good generalization performance of AdaBoost in the low noise regime. However, AdaBoost performs worse than other learning machines on noisy tasks [6, 7], such as the *iris* and the *breast cancer* benchmark data sets [5]. The present paper addresses the overfitting problem of AdaBoost in two ways. Primarily, it makes an *algorithmic* contribution to the problem of constructing regularized boosting algorithms. Secondly, it allows the user to roughly specify a hyper-parameter that controls the tradeoff between training error and capacity. This, in turn, is also appealing from a *theoretical* point of view, since it involves a parameter which controls a quantity that plays a crucial role in the generalization error bounds.

2 Boosting and the Linear Programming Solution

In this section, we briefly discuss the properties of the solution generated by standard AdaBoost and closely related Arc-GV[1], and discuss the relation to a linear programming (LP) solution over the class of base hypotheses G. Let $\{g_t(\mathbf{x}) : t = 1, \ldots, T\}$ be a sequence of hypotheses and $\boldsymbol{\alpha} = [\alpha_1 \ldots \alpha_T]$ their weights satisfying $\alpha_t \geq 0$. The hypotheses g_t are elements of a hypotheses class $G = \{g : \mathbf{x} \mapsto \{\pm 1\}\}$, which is defined by a base learning algorithm L.

The ensemble generates the label which is the weighted majority of the votes by $\operatorname{sign}(f(\mathbf{x}))$ where $f(\mathbf{x}) = \sum_t \frac{\alpha_t}{\|\boldsymbol{\alpha}\|_1} g_t(\mathbf{x})$. In order to express that f and therefore also the margin ρ depend on $\boldsymbol{\alpha}$ and for the ease of notation we define $\rho(\mathbf{z}, \boldsymbol{\alpha}) = yf(\mathbf{x})$, where $\mathbf{z} = (\mathbf{x}, y)$. Likewise we use the *normalized* margin

* This paper is a short version of [8].

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 341-344, 2000.
© Springer-Verlag Berlin Heidelberg 2000

$\rho(\boldsymbol{\alpha}) = \min_{1 \le i \le m} \rho(\mathbf{z}_i, \boldsymbol{\alpha})$. The minimization objective function of AdaBoost can be expressed in terms of margins $\mathcal{G}(\boldsymbol{\alpha}) := \sum_{i=1}^{m} \exp(-\|\boldsymbol{\alpha}\|_1 \rho(\mathbf{z}_i, \boldsymbol{\alpha}))$. In every iteration AdaBoost tries to minimize this error by a stepwise maximization of the margin. It is believed (but not proven) that AdaBoost asymptotically approximates (up to scaling) the solution of the following linear programming problem over the complete hypothesis set G

$$\begin{aligned} &\text{maximize} && \rho \\ &\text{subject to } \rho(\mathbf{z}_i, \boldsymbol{\alpha}) \ge \rho \text{ for all } 1 \le i \le m \\ &&& \alpha_t, \rho \ge 0 \text{ for all } 1 \le t \le |G| \\ &&& \|\boldsymbol{\alpha}\|_1 = 1 \ . \end{aligned} \tag{1}$$

Breiman [1] proposed a modification of AdaBoost, Arc-GV, making it possible to show the asymptotic convergence of $\rho(\boldsymbol{\alpha}^t)$ to the global solution: $\lim_{t \to \infty} \rho(\boldsymbol{\alpha}^t) = \rho^{\mathrm{lp}}$, where ρ^{lp} is the maximum possible margin for a combined classifier from G.

3 ν-Arc

Let us consider the case where we are given a (finite) set $G = \{g : \mathbf{x} \mapsto [-1, 1]\}$ of T hypotheses. To find the coefficients $\boldsymbol{\alpha}$ for the combined hypothesis $f(\mathbf{x})$ we extend the LP-AdaBoost algorithm [4, 7] and solve the following linear optimization problem, similar in spirit to [10]:

$$\begin{aligned} &\text{maximize} && \rho - \tfrac{1}{\nu m} \textstyle\sum_{i=1}^{m} \xi_i \\ &\text{subject to } \rho(\mathbf{z}_i, \boldsymbol{\alpha}) \ge \rho - \xi_i \text{ for all } 1 \le i \le m \\ &&& \xi_i, \alpha_t, \rho \ge 0 \text{ for all } 1 \le t \le T \text{ and } 1 \le i \le m \\ &&& \|\boldsymbol{\alpha}\|_1 = 1 \ . \end{aligned} \tag{2}$$

This algorithm does not force all margins to be beyond zero and we get a *soft margin* classification with a regularization constant $\frac{1}{\nu m}$. Interestingly, it can be shown that ν is asymptotically proportional to the fraction of patterns in the margin area [8].

Suppose, we have a very large base hypothesis class G. Then it is very difficult to solve (2) as (1) directly. To this end, we propose an algorithm, ν-Arc, that can approximate the solution of (2). The optimal ρ for fixed margins $\rho(\mathbf{z}_i, \boldsymbol{\alpha})$ in (2) can be written as

$$\rho_\nu(\boldsymbol{\alpha}) := \operatorname*{argmax}_{\rho \in [0,1]} \left(\rho - \frac{1}{\nu m} \sum_{i=1}^{m} (\rho - \rho(\mathbf{z}_i, \boldsymbol{\alpha}))_+ \right), \tag{3}$$

where $(\xi)_+ = \max(\xi, 0)$. Setting $\xi_i = (\rho_\nu(\boldsymbol{\alpha}) - \rho(\mathbf{z}_i, \boldsymbol{\alpha}))_+$ and subtracting $\frac{1}{\nu m} \sum_{i=1}^{m} \xi_i$ from the resulting inequality on both sides, yields (for all $1 \le i \le m$)

$$\rho(\mathbf{z}_i, \boldsymbol{\alpha}) + \xi_i \ge \rho_\nu(\boldsymbol{\alpha}) \tag{4}$$

$$\rho(\mathbf{z}_i, \boldsymbol{\alpha}) + \xi_i - \frac{1}{\nu m} \sum_{i=1}^{m} \xi_i \ge \rho_\nu(\boldsymbol{\alpha}) - \frac{1}{\nu m} \sum_{i=1}^{m} \xi_i \ . \tag{5}$$

In particular we have to get rid of the slack variables ξ_i again by absorbing them into quantities similar to $\rho(\mathbf{z}_i, \boldsymbol{\alpha})$ and $\rho(\boldsymbol{\alpha})$. This works as follows: on the right

hand side of (5) we have the objective function (cf. (2)) and on the left hand side a term that depends nonlinearly on $\boldsymbol{\alpha}$. Defining

$$\tilde{\rho}_\nu(\boldsymbol{\alpha}) = \rho_\nu(\boldsymbol{\alpha}) - \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i, \quad \tilde{\rho}_\nu(\mathbf{z}_i,\boldsymbol{\alpha}) = \rho(\mathbf{z}_i,\boldsymbol{\alpha}) + \xi_i - \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i, \tag{6}$$

which we substitute for $\rho(\boldsymbol{\alpha})$ and $\rho(\mathbf{z},\boldsymbol{\alpha})$ in (1), respectively, we obtain a new optimization problem. Note that $\tilde{\rho}_\nu(\boldsymbol{\alpha})$ and $\tilde{\rho}_\nu(\mathbf{z}_i,\boldsymbol{\alpha})$ play the role of a *corrected* or *virtual* margin. We obtain a non-linear min-max problem in terms of $\tilde{\rho}$

$$\begin{aligned} &\text{maximize} && \tilde{\rho}(\boldsymbol{\alpha}) \\ &\text{subject to } \tilde{\rho}(\mathbf{z}_i,\boldsymbol{\alpha}) \geq \tilde{\rho}(\boldsymbol{\alpha}) \text{ for all } 1 \leq i \leq m \\ &&& \alpha_t \geq 0 \text{ for all } 1 \leq t \leq T \\ &&& \|\boldsymbol{\alpha}\|_1 = 1\ , \end{aligned} \tag{7}$$

which we refer to as *ν-Arc*.

We can now state interesting properties for ν-Arc by using Theorem 5 of [9] that bounds the generalization error $R(f)$ for ensemble methods. In our case $R_\rho(f) \leq \nu$ by construction, thus we get the following simple reformulation of this bound:

$$R(f) \leq \nu + \sqrt{\frac{c}{m}\left(\frac{h\log^2(m/h)}{\rho_\nu^2} + \log\left(\frac{1}{\delta}\right)\right)}\ . \tag{8}$$

The tradeoff in minimizing the right hand side between the first and the second term is controlled directly by an easy interpretable regularization parameter ν.

4 Experiments

We show a set of toy experiments to illustrate the general behavior of ν-Arc-GV. As base hypothesis class G we use RBF networks [7], and as data a two-class problem generated from several 2D Gauss blobs. We obtain the following results:

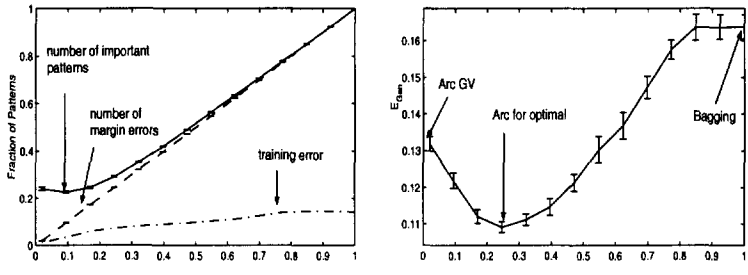


Fig. 1. Toy experiment: the left shows the average fraction of *important* patterns, the average fraction of margin errors and the average training error for different values of the regularization constant ν for ν-Arc. The bottom plots show the corresponding generalization error. The parameter ν allows us to reduce the test errors to values about 20% (relatively) lower than for the hard margin algorithm (for $\nu = 0$ we recover Arc-GV/AdaBoost and for $\nu = 1$ we get Bagging.)

- ν-Arc leads to approximately νm patterns that are effectively used in the training of the base learner: Figure 1 (left) shows the fraction of patterns that have high average weights during the learning process.
- ν-Arc leads to the fraction ν of margin errors (cf. dashed line in Figure 1) exactly.
- The (estimated) test error, averaged over 10 training sets, exhibits a rather flat minimum in ν (Figure 1 (right)).

5 Conclusion

We analyzed the AdaBoost algorithm and found that Arc-GV and AdaBoost are suitable for approximating the solution of non-linear min-max problems over huge hypothesis classes. We introduced a new regularization constant ν that controls the fraction of patterns inside the margin area. The new parameter is highly intuitive and has to be tuned only within a fixed interval $[0, 1]$.

We found empirically that the generalization performance in ν-Arc is robust against changes around the optimal choice of the regularization parameter ν. This finding makes model selection (e.g. via cross-validation) much easier.

As the patterns in the margin area correspond to interesting, difficult and informative patterns, future research will focus on using Boosting and Support Vector methods for data mining purposes.

References

1. L. Breiman. Prediction games and arcing algorithms. Technical Report 504, Statistics Department, University of California, December 1997.
2. H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 7:705 – 719, 1993.
3. Y. LeCun et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks*, pages 261–276, 1995.
4. A. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proc. of the 15th Nat. Conf. on AI*, pages 692–699, 1998.
5. C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1998. [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA.
6. J. R. Quinlan. Boosting first-order learning (invited lecture). *Lecture Notes in Computer Science*, 1160:143, 1996.
7. G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. Technical Report NC-TR-1998-021, NeuroColt, 1998. To appear in Machine Learning.
8. G. Rätsch, B. Schökopf, A. Smola, S. Mika, T. Onoda, and K.-R. Müller. Robust ensemble learning. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 1999.
9. R. Schapire, Y. Freund, P. L. Bartlett, and W. Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.
10. B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1083 – 1121, 2000.
11. H. Schwenk and Y. Bengio. Training methods for adaptive boosting of neural networks. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Inf. Processing Systems*, volume 10. The MIT Press, 1998.

Interactive Visualization in Mining Large Decision Trees

Trong Dung Nguyen, Tu Bao Ho, and Hiroshi Shimodaira

Japan Advanced Institute of Science and Technology
Tatsunokuchi, Ishikawa, 923-1292 JAPAN

Abstract. This paper presents a tree visualizer that combines several techniques from the field of information visualization to handle efficiently large decision trees in an interactive mining system.

1 Introduction

Research on visualization of decision trees has recently received a great attention from the KDD (knowledge discovery and data mining) community because of its practical importance. Many works have been done, e.g., the 3D Tree Visualizer in system MineSet [2], CAT scan (classification aggregation tablet) for inducing bagged decision trees [5], the interactive visualization in decision tree construction [1], etc. In our development of a tree visualizer for the system CABRO [4], a KDD system based on decision tree induction, we face two problems of decision tree visualization that have not thoroughly investigated: integration of tree visualization into an interactive knowledge discovery process and visualization of large trees.

This paper represents the tree visualizer of CABRO, which can be invoked during the discovery process and can handle large trees by combining several techniques in information visualization. We also introduce a new technique, called T2.5D, for large tree visualization.

2 The Tree Visualizer

This section describes an overview of the tree visualizer in CABRO. The tree visualizer of CABRO provides several *views*; each view uses variant techniques and drawing algorithms to display trees, and serves different usage purposes. The available views are:

- **Standard**: The tree is drawn in proportion, the size of a node is up to the length of its label, a father is vertically located at the middle of its children, and sibling are horizontally aligned.
- **Tightly-coupled** [3]: The window is divided into two panels, one displays the tree in a tiny size and the other displays it in a normal size. The user uses the first panel as a map to navigate, and sees the focused area in the second one.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 345–348, 2000.
© Springer-Verlag Berlin Heidelberg 2000

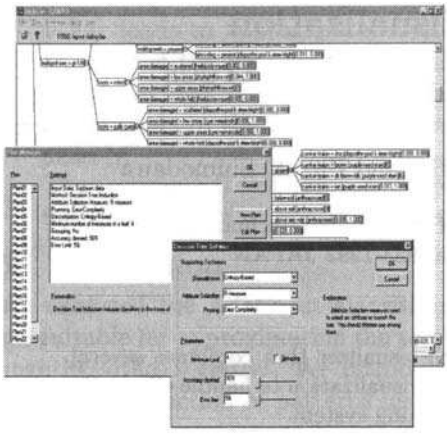


Fig. 1. Model selection

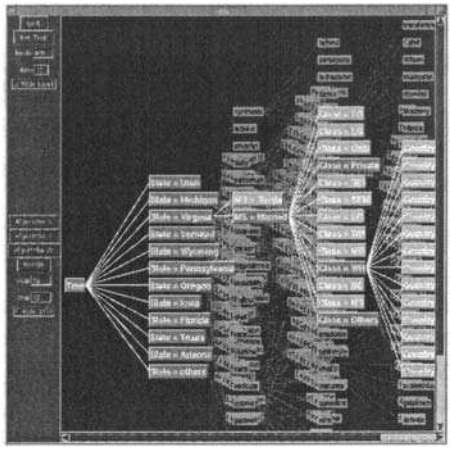


Fig. 2. The T2.5D view

- **Fish-eyes** [3]: This view distorts the magnified image so that nodes around the center of interest is displayed at high magnification, and the rest of the tree is progressively compressed.
- **T2.5D**: This view is based on our proposed technique named T2.5D (stands for Tree 2.5 Dimensions) that tries to save screen space in order to display more nodes in a compact window. The focused paths of the tree are drawn in the front and in highlighted colors, the other paths are in background and dim colors (Fig. 2).

The tree visualizer allows the user to customize above views by applying several *operations* on them. These operations include:

- **Zoom**: The user can zoom in or zoom out the drawn tree.
- **Collapse/expand**: The user can choose to view some parts of the tree by collapsing and expanding paths.
- **Display node**: The user can see the content of a node such as attribute/value, class, population, etc.

For different purposes of usage, the tree visualizer provides three *modes* of use:

- **Viewing**: In this mode, the user takes whole control of the visualizer; he/she is free to choose which and how trees to be displayed. This mode is used when the user just wants to view a particular tree.
- **Mining**: In this mode, the system automatically invokes tree visualizer to display trees under consideration. For example, changes of a generating tree are continuously displayed, so the user can see the progress of induction.
- **Matching**: In this mode, the tree visualizer shows visual feedback for matching unknown objects, such as highlighting matched paths and matched objects or graying out the unmatched parts of the tree.

3 Interactive Discovery and Large Decision Trees

This section analyses the role of the tree visualizer in the interactive discovery process of CABRO, and how it can handle efficiently with trees of different sizes.

3.1 Model Selection

In CABRO, to construct decision trees, the user needs to choose a series of settings including which attribute selection and pruning techniques to be used, what is the minimum number of instances at a leaf node, what is the lowest accepted error rate, whether the values of an attribute will be grouped, how to deal with missing values, etc. Furthermore, if there are some continuous attributes the user may have to choose some discretization technique to discretize them. The chosen settings will be registered in a form called a plan. The realization of a plan will yield a decision tree (model). As there are many possible combinations of settings, the user usually has to try a number of plans to achieve appropriate decision trees. We consider that work *model selection.* After decision trees are generated, the user can visualize, evaluate and compare them. This iterative cycle may be repeated until reaching appropriate models. Usually, the model selection is a daunting and very time consuming work, but thanks to the tree visualizer and the flexible user interface of the system, it becomes easier and more effective (Fig. 1).

The system frequently invokes the tree visualizer in the interaction with the user throughout model selection whenever he/she needs to refer to trees in order to make a choice. The tree visualizer then is in mining mode; it automatically chooses an arrangement of views that may be most convenient for the user to get needed information. Moreover, if the induction is set to run interactively, the system allows the user to be able to take part at every step of the induction, for example, he/she can manually choose which attribute will be used to branch a node, or which branch has to be pruned. The tree visualizer then will display candidate trees corresponding to different possibilities of the tree construction. The user then can decide which one is promising to further construct.

3.2 Large Decision Trees

There are many researches in the field of information visualization for representing large hierarchical structures, e.g., treemap, cone trees, hyperbolic trees [3], however they are not directly applicable to decision tree visualization. In the tree visualizer of CABRO, we deal with large trees by combining many known techniques whenever they fit and developing a the T2.5D technique. Those techniques are used to implement the views and operations. They complement each other to give the user a convenient visualization in different situations in discovering and using decision trees. For example, the standard view is the best for small and average trees as it displays a very proportional picture of tree structure. Used together with the collape/expand operation the standard mode can deal well with bigger trees. Tightly-coupled views help the user to locate easily

any area in the tree picture, while fish-eye view is the best if the user currently focuses on a single node but also wants to see clearly its neighbors.

Very large trees are still difficult to navigate and view even with tightly-coupled and fish-eye techniques. To solve the problem, we have been developing the technique T2.5D. The 3D browsers usually can display more nodes in a compact area of the screen but require currently expensive 3D animation support and the structure somehow not easy to navigate, while the 2D browsers have a limitation in displaying many nodes in one view. The T2.5D technique combines the advantages of both the 2D and 3D drawing techniques to provide the user an efficient view with lower processing cost. The T2.5D view can display more than 1000 nodes in full size (some are partially overlapped). In T2.5D, a node can be highlighted or dim. The highlighted nodes are those the user currently pays most attention on, and they are displayed in 2D for ease of view and navigation. The dim nodes are displayed in 3D to save the space, they allow the user to get an idea about overall structure of the hierarchy.

4 Conclusion

We addressed problems of integration of tree visualization into an interactive knowledge discovery process, and visualization of large trees. We presented the interactive tree visualizer of the system CABRO that employs recent visualization techniques in mining decision trees, and how it works in an interactive discover process. We described our attempt to deal with large tree by implementing multiple complemented views and operations, and introducing the new technique T2.5D. Though there are still a lot of work for improving this interactive tree visualizer, we believe that it contributes an efficient solution to the state-of-the-art of visualization of decision trees in KDD, especially T2.5D is very promising as a new display and navigation technique for large trees.

References

1. Ankerst, M., Elsen, C., Ester, M., Kriegel, H. P.: Visual Classification: An Interactive Approach to Decision Tree Construction. *Proc. of Fifth Inter. Conf. on Knowledge Discovery and Data Mining* (1999), 392–397.
2. Brunk, C., Kelly, J. and Kohavi, R.: MineSet: An Integrated System for Data Mining. *Proc. of Third Inter. Conf. on Knowledge Discovery and Data Mining* (1997) 135–138.
3. Card, S., Mackinlay, J., Shneiderman B.: *Readings in Information Visualization: Using Vision To Think*, Morgan Kaufmann Publishers, Inc. (1999).
4. Nguyen, D. T., Ho, T. B.: An Interactive-Graphic System for Decision Tree Induction. *Journal of Japanese Society for Artificial Intelligence*, Vol. 14 (1999) 131–138.
5. Rao, J. S., Potts, W. J. E.: Visualizing Bagged Decision Trees. *Proc. of Third Inter. Conf. on Knowledge Discovery and Data Mining*, AAAI Press (1997) 243–246.

VQTree: Vector Quantization for Decision Tree Induction

Shlomo Geva and Lawrence Buckingham

Queensland University of Technology,
GPO Box 2434, Brisbane QLD 4001, Australia

Abstract. We describe a new oblique decision tree induction algorithm. The VQTree algorithm uses Learning Vector Quantization to form a non-parametric model of the training set, and from that obtains a set of hyperplanes which are used as oblique splits in the nodes of a decision tree. We use a set of public data sets to compare VQTree with two existing decision tree induction algorithms, C5.0 and OC1. Our experiments show that VQTree produces compact decision trees with higher accuracy than either C5.0 or OC1 on some datasets.

1 Introduction

Decision trees are widely used for classification problems [Breiman *et al.* 1984], [Quinlan 1992], [Murthy *et al.* 1994]. Their popularity is easy to understand, as they provide efficient, readily interpreted models of the input-output mapping for a given classification task.

Conceptually, a decision tree is a branching structure of linked nodes, which taken together represent a set of nested *if-then-else* tests. The two main kinds of decision tree are commonly referred to as *axis-parallel* and *oblique* trees. Axis-parallel trees are characterized by decisions of the form

$$attribute = value$$

used for attributes with symbolic values, or

$$attribute \leq value$$

used for numeric attributes. Examples of axis-parallel decision tree systems are CART [Breiman *et al.* 1984] and C4.5 [Quinlan 1992]. The decisions used in oblique trees are linear inequalities involving more than one attribute:

$$\boldsymbol{w}^T \boldsymbol{x} \leq value,$$

where $\boldsymbol{x}$ is a vector of attribute values and $\boldsymbol{w}$ is a weight vector. Examples of oblique decision tree systems are Linear Machine Decision Trees (LMDT) [Utgoff & Brodley 1991] and OC1 [Murthy *et al.* 1994].

Quinlan observes that some classification tasks are inherently less well suited for axis-parallel decision trees than others [Quinlan 1994]. When the decision

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 349–359, 2000.
© Springer-Verlag Berlin Heidelberg 2000

surface involves few interactions between attributes we expect axis-parallel trees to be very successful. Conversely, if the decision surface involves interactions between multiple attributes then oblique hyperplanes provide a more economical and accurate representation of the decision surface [Murthy *et al.* 1994].

Decision trees are usually built by a recursive divide-and-conquer process which is very efficient for axis-parallel decision trees [Quinlan 1992, page 17]. Suppose there are n examples, each of which has d attributes. Then there are a maximum of n possible distinct binary splits for each attribute, so that the maximum number of axis-parallel binary splits is just $n \times d$. Therefore it is feasible to exhaustively test all possible splits at each branch [Quinlan 1992, page 25]. However, there may be as many as $2^d\binom{n}{d}$ distinct oblique binary splits of the same set of examples [Murthy *et al.* 1994, page 3]. To get around this, oblique decision tree algorithms must constrain the set of oblique tests to be considered. This is usually done by a sampling or search procedure.

In the present paper we suggest another approach, the VQTree algorithm, that combines the strengths of both approaches. The algorithm uses a preprocessing step to create a pool of oblique hyperplanes called the *decision set*, which contains a piecewise linear approximation to the decision surface. Then standard top-down induction is applied to the training set, using hyperplanes from the decision set to subdivide the training subset at each node. The use of oblique splits lets us take advantage of naturally appearing linear interactions in the training set. Use of a preprocessing step to identify candidate splits allows us to exhaustively evaluate the splits during the induction step.

The remainder of the paper is laid out as follows. In Section 2 we introduce decision tree induction and provide brief descriptions of several existing oblique decision tree systems. Section 3 describes the VQTree algorithm in detail. In Section 4 we outline a set of experiments which verify the utility of the algorithm. Finally in Section 5 we summarize and point the way for further exploration.

2 Previous Work on Oblique Decision Tree Induction

We consider classification problems, in which there is an input space $\mathcal{X} \subseteq \Re^D$, a discrete set of class labels $\mathcal{C}$, and an unknown relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{C}$. We have a (given) training set of labelled examples $T \subset \mathcal{R}$, which are assumed to be drawn according to fixed but unknown probability distribution $\mathcal{D}$ defined over $\mathcal{R}$. The goal is to find a function $F : \mathcal{X} \rightarrow \mathcal{C}$ which approximates $\mathcal{R}$ in such a way that the probability of classification error is minimal with respect to $\mathcal{D}$.

2.1 Top-Down Decision Tree Induction

In a decision tree system, F is implemented as a nested series of multi-way *if-then-else* tests, which subdivide $\mathcal{X}$ into a disjoint set of polyhedral decision regions. Each leaf of the decision tree has a class label and covers one decision region. A point is classified by traversing the tree to determine which region it lies in, and taking the class label of the associated leaf. Decision trees are built

by a greedy divide and conquer method in which the training set is recursively partitioned into subsets, controlled by a "goodness of split" measure, G.

At stage i of construction, a pool of candidate tests S_i is available, as well as a set T_i of available training points. Each test $s \in S_i$ is used to split T_i into subsets $\{T_{i1}(s), ...T_{in}(s)\}$, and a measure of quality is assigned to the resulting partition using G. The test s^* that yields the best quality measure is chosen for the current branch, s^* is removed from the pool of candidate tests, and the process is applied recursively to each subset $T_{ij}(s^*)$ to produce the decision tree. Branching ceases when all examples in a subset belong to the same class, or when some alternative stopping criterion is met. Once a test has been chosen it is not reviewed during the construction phase. Many algorithms also employ a pruning stage, when redundant nodes are eliminated to improve the accuracy of the decision tree.

Several metrics have been used to measure goodness of split. Information Gain, the method used by C4.5, has proved to be robust and effective [Quinlan 1992, page 21]. However, a number of other measures are detailed in [Murthy *et al.* 1994, Appendix B].

2.2 Oblique Decision Tree Algorithms

LMDT [Utgoff & Brodley 1991] constructs a *linear machine* in each branch, consisting of a set of weight vectors $\{\boldsymbol{w}_c | c \in C\}$, one for each class. The weight vectors are optimized by an iterative process in which training points are presented and reward or punishment steps are applied to each weight vector. If training point $(\boldsymbol{x}, y)$ is presented, the update step is

$$\boldsymbol{w}_c \leftarrow \boldsymbol{w}_c \pm \rho \boldsymbol{x}.$$

A positive update occurs if and only if the classes match, ie. $c = y$. As training continues the correction factor ρ decays according to the schedule:

$$\beta \leftarrow 0.995\beta - 0.0005$$
$$\rho \leftarrow \frac{\beta^2}{1+\beta}.$$

Since the linear machine in each node contains one weight vector for each class, LMDT produces multi-way splits with oblique hyperplanes.

OC1 [Murthy *et al.* 1994] is derived from a variant of CART which used linear combinations of attributes in tests, and employed a randomized search method [Breiman *et al.* 1984, chapter 5]. OC1 places a single hyperplane in each branch, forming an oblique binary split. The hyperplane is initialized with the best axis-parallel binary split for the current subset of training examples. After initial placement, a random hill-climbing method is used to improve the quality of the split. Since the worst that can happen during the hill-climbing stage is "no improvement", the resulting oblique test will be at least as good as the best axis-parallel binary split.

[Sanchez *et al.* 1998] describe a method for constructing a decision tree from the Voronoi diagram of the training set. The Voronoi diagram is a set of polyhedral cells, one for each point in the training set. The polyhedron surrounding a training point contains the set of all points closer to that training point than any other point in the training set, using the Euclidean metric to measure distances. Thus the task of determining the nearest training point to a query point is equivalent to finding which polyhedron in the Voronoi diagram contains the query point. The cell boundaries in the Voronoi diagram form a set of hyperplanes ∇, which are used to construct an oblique decision tree. The algorithm is:

- Obtain the decision hyperplanes ∇ from the training set.
- Recursively split $\Re^D$ with hyperplanes from ∇ to form a decision tree.
- Assign class labels to each leaf of the decision tree:
  - Generate a query point that lies in the region covered by the leaf.
  - Search the training set to find the nearest training point to the query point.
  - The class label of the nearest training point becomes the class represented by the leaf.
- Prune the tree by merging any sibling leaves that represent the same class.

The VQTree algorithm differs in several practical ways from all of these methods. Like the algorithm of [Sanchez *et al.* 1998], we do not recursively subdivide the training set during the search for candidate tests, and hyperplanes are obtained from a Voronoi diagram. However, by conducting Vector Quantization as the first step we obtain a compressed representation of the underlying data. This reduces the number of candidate splits that must be evaluated during the top-down induction step.

3 The VQTree Algorithm

The VQTree algorithm consists of the following steps:

1. Select an initial nearest neighbour codebook from the training set.
2. Optimize the nearest neighbour codebook using one of the algorithms in the Learning Vector Quantization (LVQ) family [Kohonen 1988].
3. Derive a set of hyperplanes (called the *decision set*) from the codebook by examining the adjacency relationships between prototypes.
4. Postprocess the training set to reduce classification noise.
5. Build a decision tree using top-down induction as described in Section 2.

3.1 Selection of Initial Codebook

Input: Training set, $T \subset \mathcal{R}$
Number of prototypes required, $m^* > 0$
Output: Codebook, $\mathcal{M} \subset \mathcal{R}$

A *codebook* is a set $\mathcal{M}$ of labelled *prototype points* belonging to the same space as the target relation $\mathcal{R}$. Thus, $\mathcal{M} = \{(\mu_1, \phi_1), ..., (\mu_m, \phi_m)\} \subset \mathcal{X} \times \mathcal{C}$. The codebook is used to classify a point $\boldsymbol{x}$ by finding the nearest prototype μ_{j^*} in $\mathcal{M}$, and taking the associated class ϕ_{j^*}. Distances are calculated using the Euclidean metric.

We use a variant of the *random coding* method [Gersho & Gray 1992, page 359] to seed the codebook with approximately m randomly chosen training points. Class prior probabilities are taken into account to give a balanced spread of prototypes across all classes. The number of prototypes drawn for each class depends upon the total number required and on the relative proportions of the classes in the training set. If class c has frequency f_c in the training set while the training set has N points, then the number of prototypes drawn for that class is $m_c = \lceil m^* \times \frac{f_c}{N} \rceil$. The complete codebook will contain $m = \sum_{c \in \mathcal{C}} m_c$ labelled prototypes.

3.2 Codebook Optimization

Input: Training set, $T \subset \mathcal{R}$
Codebook, $\mathcal{M} \subset \mathcal{R}$
Output: Optimised codebook, $\mathcal{M}' \subset \mathcal{R}$

Kohonen's LVQ algorithms [Kohonen 1988] optimize nearest neighbour codebooks for classification. They are computationally efficient, readily able to handle very large training sets. Although there are some differences in detail between LVQ algorithms, the central point is the learning step, which is common to all. At each time step t, a training example $\boldsymbol{x}(t)$ is used to update one or more prototypes $\mu(t)$ using a reward/punishment regime of the form:

$$\mu(t) \leftarrow \mu(t) \pm \alpha(t)(\boldsymbol{x}(t) - \mu(t)).$$

Here $\alpha(t)$ is a learning rate, which may vary during training. The update is positive if and only if the class of point $\boldsymbol{x}(t)$ and prototype $\mu(t)$ are the same. If the training example $\boldsymbol{x}(t)$ contains missing values, then distances are calculated and updates are applied in the subspace of non-missing values.

The chief difference between LVQ algorithms is the criterion used to decide which (if any) prototype $\mu(t)$ to adjust when presented with $\boldsymbol{x}(t)$. For our experiments here, we have used LVQ1 [Kohonen 1988] and the Decision Surface Mapping (DSM) algorithm [Geva & Sitte 1991].

LVQ1 forms a non-parametric model of the distribution of training points within each class, and implicitly approximates the Bayes decision surface. LVQ1 is fairly robust in the presence of noise in the training set. Under LVQ1, when a training point $\boldsymbol{x}(t)$ is presented, the single nearest prototype is adjusted.

DSM strives to form an accurate model of the Bayes decision surface at the expense of quantization within each class. When point $\boldsymbol{x}(t)$ is presented, one of two actions may be taken. If the nearest neighbour to $\boldsymbol{x}(t)$ in the codebook has the same class label as $\boldsymbol{x}(t)$ then no update occurs. However, if the nearest neighbour to $\boldsymbol{x}(t)$ has a different class label to $\boldsymbol{x}(t)$, two prototypes are adjusted:

- the actual nearest prototype is pushed away from $\boldsymbol{x}(t)$, because its class is wrong.
- the nearest prototype with the right class is pulled towards $\boldsymbol{x}(t)$.

Because updates occur whenever a training example is misclassified, DSM is subject to problems when the data is noisy. However, if the conditions are right DSM can produce classifiers that are considerably more accurate and compact than those trained with LVQ1.

3.3 Extract the Decision Set

Input: Training set, $T \subset \mathcal{R}$
Codebook, $\mathcal{M} \subset \mathcal{R}$
Output: Decision set S, a set of hyperplanes

Once the codebook has been created, each prototype $\boldsymbol{\mu}_j$ is surrounded by a decision region which includes the points nearer to $\boldsymbol{\mu}_j$ than to any other prototype. All points within this region will be mapped to the class ϕ_j by the nearest neighbour classifier. If $\boldsymbol{\mu}_j$ and $\boldsymbol{\mu}_k$ are two adjacent prototypes then the hyperplane H_{jk} which separates their decision regions has an equation of the form

$$\begin{aligned} H_{jk}(\boldsymbol{x}) &= \boldsymbol{w}_{jk}^T \boldsymbol{x} + b_{jk} = 0 \\ \text{where } \boldsymbol{w}_{jk} &= \boldsymbol{\mu}_j - \boldsymbol{\mu}_k \\ \text{and } b_{jk} &= -\frac{1}{2}(\boldsymbol{\mu}_j + \boldsymbol{\mu}_k)^T \boldsymbol{w}_{jk}. \end{aligned}$$

We are interested only in the set of hyperplanes that separate adjacent decision regions mapping to different classes,

$$S \subseteq \{H_{jk} | 1 \leq j, k \leq m \wedge \phi_j \neq \phi_k\}.$$

One way to find S would be to first compute the Voronoi diagram of the codebook, and from there read off the hyperplanes. However, computing the Voronoi diagram in high dimensional space is a hard problem since there may be an exponential number of vertices. Since we do not need to know the vertices of the diagram, we are able to tackle a simpler problem, namely to determine whether or not two prototypes have adjacent decision regions. In the VQTree prototype this is done using a polynomial-time algorithm based on ray-casting, although it could also be formulated as a linear programming task [Sanchez *et al.* 1998].

The algorithm we use to determine whether the hyperplane H_{jk} is to be included in S is:

1. $\boldsymbol{x} \leftarrow \boldsymbol{\mu}_j, \boldsymbol{y} \leftarrow \boldsymbol{\mu}_k$.
2. $\mathcal{A} \leftarrow \{H_{ji} | i \neq j\}, \mathcal{U} \leftarrow \{\}$.
3. $\forall H_{ji} \in \mathcal{A}, \alpha_i \leftarrow (\boldsymbol{w}_{ji}^T \boldsymbol{x} + b_{ji})/(\boldsymbol{w}_{ji}^T \boldsymbol{x} + \boldsymbol{w}_{ji}^T \boldsymbol{y})$.
4. $p \leftarrow \arg\min_i \{\alpha_i | \alpha_i \geq 0\}$.
5. If $p = k$, then signal success and STOP.
6. $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha_p(\boldsymbol{y} - \boldsymbol{x})$

7. $\mathcal{A} \leftarrow \mathcal{A} - \{H_{jp}\}, \mathcal{U} \leftarrow \mathcal{U} \cup \{H_{jp}\}$
8. If $H_{ji}(\boldsymbol{x}) < 0$ for some $H_{ji} \in \mathcal{U}$, then signal failure and STOP. (The hyperplanes are assumed to be oriented so that $H_{ji}(\boldsymbol{x}) > 0$ if $\boldsymbol{x}$ is within the decision region.)
9. $\boldsymbol{y} \leftarrow \boldsymbol{y} - \left(\boldsymbol{w}_{jp}^T \boldsymbol{\mu}_j + b_{jp}\right) / \left(\boldsymbol{w}_{jp}^T \boldsymbol{w}_{jp}\right) \boldsymbol{w}_{jp}$.
10. Go to step 3.

The codebook contains m prototypes, so there are $O(m^2)$ possible hyperplanes to consider. For each of these hyperplanes, step 3 requires up to $O(m^2)$ intersections to be calculated. As each intersection requires the computation of several dot-products between vectors, we obtain an upper bound of $O(Dm^4)$ for the complexity of this algorithm. Thus, clear advantages are to be had if the number of prototypes is kept small.

3.4 Reduce Classification Noise in Training Set

Input: Training set, $T \subset \mathcal{R}$
Codebook, $\mathcal{M} \subset \mathcal{R}$
Output: "Clean" training set, $T' \subset \mathcal{R}$

During implementation of this algorithm we encountered the practical difficulty that, if the training set contains many points with a class label that does not match the value predicted by the nearest neighbour classifier, then the decision tree overfits the noisy data. This problem seems to stem from the fact that, during early splits especially, hyperplanes are applied to split training data far from the region where they actually appear as part of the decision surface.

To get around this difficulty, we perform a postprocessing step to clean up class noise in the training set. The class label of each training point is replaced with a new label, obtained by nearest neighbour search in the codebook.

3.5 Top-Down Induction and Pruning

Input: "Clean" training set, $T \subset \mathcal{R}$
Decision set S, a set of hyperplanes
Output: Oblique decision tree

Having obtained the decision set, it remains only to apply the top-down induction algorithm described in Section 2 to complete the decision tree. For training points with missing values, we determine the position relative to each hyperplane by comparing the distance from the training point to each of the prototypes that generated the hyperplane, restricting the calculation to the subspace of non-missing values. After the tree has been grown, we perform a bottom-up pruning pass, which identifies any branch that has two leaves with the same class label. If such a branch is found, it is replaced by a single leaf representing the common class. This form of pruning simplifies the tree, but has no effect on the classification accuracy.

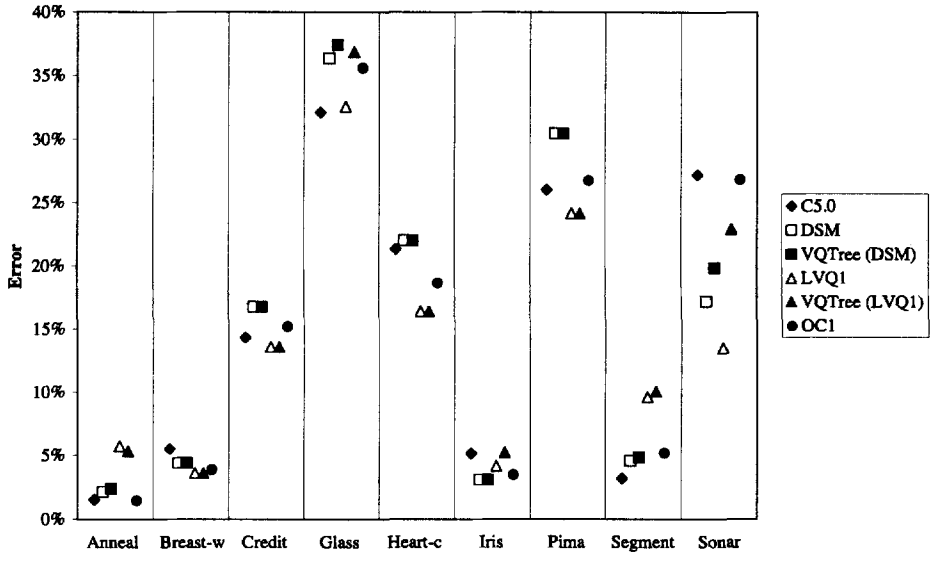


Fig. 1. Test error on 9 benchmark datasets.

4 Experimental Results

This section describes the results of a set of tests, comparing VQTree with C5.0 Release 1 (the commercial version of C4.5), and with OC1 Release 3. We employed two LVQ algorithms (DSM and LVQ1) to optimise the nearest neighbour codebook, yielding two variants of VQTree classifier which we term VQTree (DSM) and VQTree (LVQ1) respectively. We tested the algorithms on 9 data sets from the UCI Machine Learning Repository [Murphy & Aha 1994], chosen primarily because they include a large proportion of continuous attributes. For each data set, we conducted 10 repeats of 10-fold cross validation, presenting the same training and test sets to all algorithms at each stage.

Minimal preprocessing was conducted before we started the experiments. For the Anneal domain, missing values were converted to valid "not applicable" symbolic values. Other than that, for C5.0 the data was used as-is from the repository. VQTree and OC1 require real-valued attributes, so as each training-testing partition was generated we converted symbolic values to sparse-coded binary values and standardised all attributes using the transformation

$$x_i \leftarrow \frac{(x_i - \overline{x}_i)}{\sigma_i}.$$

Here x_i is attribute i, $\overline{x}_i$ is the mean of x_i and σ_i is the standard deviation of x_i, computed using values from the training set.

The optimal number of prototypes to include in the codebook is not usually known before a classifier is built, so we ran a series of experiments using 2, 4, 8,

Table 1. *Pairwise combinations of the 4 decision trees involved in this study. Each cell contains the number of wins, draws and losses between the decision tree in that row versus the decision tree in that column.*

Algorithm	C5.0	OC1	VQTree (DSM)	VQTree (LVQ1)
C5.0	–	4 – 2 – 3	5 – 1 – 3	3 – 1 – 5
OC1	3 – 2 – 4	–	5 – 2 – 2	3 – 2 – 4
VQTree (DSM)	3 – 1 – 5	2 – 2 – 5	–	4 – 1 – 4
VQTree (LVQ1)	5 – 1 – 3	4 – 2 – 3	4 – 1 – 4	–

16, 32 and 64 prototypes per codebook[1], reporting the best result for each data set. DSM and LVQ1 were trained for 20 epochs, with an initial learning rate $\alpha(t)$ of 0.1, decaying linearly to 0. C5.0 and OC1 were run using default parameter values, so pruned trees were produced.

Figure 1 shows the test error of the four decision trees and the underlying nearest neighbour classifiers from which each VQTree classifier was derived. Thus, the series "DSM" refers to the nearest neighbour classifier trained with DSM, while the series "VQTree (DSM)" refers to the corresponding decision tree. The two VQTree algorithms perform well compared to C5.0 and OC1, with the LVQ1 variant attaining the lowest test error on 4 of the 9 datasets while the DSM variant reached the lowest error on 2. C5.0 attained the lowest error on 2, while OC1 was best on 1 domain.

Table 1 provides comparisons between each pair of decision trees that we tested, using 95% confidence intervals to determine significance. Within each pair, an algorithm scores a win if its test error is significantly lower than the other, a loss if the test error is significantly higher, and a draw otherwise. Again the LVQ1 variant of VQTree makes a strong showing relative to C5.0 and OC1, but it appears to be evenly matched with the DSM variant of VQTree.

Turning to the issue of classifier size, Table 2 sets out details of the optimal LVQ codebook sizes and the number of leaves in each decision tree. OC1 is the most economical of the decision tree algorithms, followed by VQTree (DSM), VQTree (LVQ1) and C5.0. The size of the classifier produced by the LVQ1 variant of VQTree is usually smaller than the original LVQ codebook, but this is not always the case for the DSM variant. This reflects the fact that LVQ1 tends to spread prototypes evenly through the training set, while DSM is more parsimonious with prototypes and places them nearer to the decision surface. Examining the optimal number of prototypes for each data set, we see that there is no hard and fast rule – for some tasks a large codebook is required, producing correspondingly complex trees. However, for other data sets a very simple oblique decision tree is best – Cleveland Heart Disease and Australian Credit data sets require only a single hyperplane.

[1] The actual number of prototypes varies slightly from these ideal values. See Section 3.1 for details.

Table 2. Average classifier size – number of leaves per decision tree, number of prototypes per nearest neighbour classifier.

Dataset	C5.0	OC1	DSM	VQTree (DSM)	LVQ1	VQTree (LVQ1)
Anneal	23.7	6.5	65	24.3	65	20.9
Breast-w	12.6	2.9	2	2.0	8	3.6
Credit	18.5	3.1	2	2.0	2	2.0
Glass	23.8	12.1	10	19.7	34	32.4
Heart-c	20.1	3.6	2	2.0	2	2.0
Iris	4.8	3.2	3	3.0	9	6.4
Pima	22.4	6.3	4	4.4	4	5.4
Segmentation	41.4	30.5	35	60.9	66	84.7
Sonar	14.4	5.8	16	22.5	64	20.6

Finally, we observe that on two datasets, Glass and Sonar, neither VQTree variant lived up to the full promise of the underlying nearest neighbour classifier. Although the decision trees do not generalise as well as the underlying classifiers, they perform identically on the training set. The large number of prototypes needed in the LVQ1 codebook for Sonar suggests that there may be many small clusters in the dataset. Since the hyperplanes of the decision set are obtained by a local process, it may be that by using them in a global manner the algorithm obscures the fine structure of the training set. This is a subject for future investigation.

5 Conclusions

In the present paper we have described an oblique decision tree induction algorithm based on Learning Vector Quantization. The algorithm identifies a set of hyperplanes, the decision set, that locally approximate the decision surface as embodied in a training set. Each of these hyperplanes gives a linear inequality which can be used to subdivide the input space into two half spaces. In a postprocessing step, an oblique tree is constructed from the decision set, using standard top-down tree induction.

To establish the viability of the algorithm, we have implemented a prototype which uses a ray-casting algorithm to build the decision set in polynomial time, and which employs the Information Gain criterion to assess the quality of a split. The prototype has been applied to several benchmark datasets from the UCI repository, yielding very encouraging results. VQTree compares favourably with C5.0 and OC1 in terms of classification accuracy, although OC1 produces more compact trees.

Despite the success of the prototype system tested here, several aspects of the system will benefit from further research, chiefly:

- A mechanism to automatically determine optimal LVQ codebook size.

- A way to determine which is the best LVQ training algorithm to employ.
- Combining oblique splits as produced by VQTree with axis-parallel splits to produce a more robust and effective system.
- Combining oblique splits derived from diverse codebooks to produce single decision trees.

Vector quantization is fairly mature classification and signal processing technology, but LVQ classifiers are usually regarded as "black-boxes" which perform an input-output mapping. In this paper we have illustrated a way in which the internal structure of the LVQ codebook can be used to advantage to produce an accurate, economical decision tree.

References

[Breiman *et al.* 1984] Breiman, L., Friedman, J.H., Olshen R.A. & Stone, C.J: *Classification And Regression Trees.* Pacific Grove, CA: Wadsworth & Brooks/Cole

[Gersho & Gray 1992] Gersho, A. & Gray, R.M.: *Vector Quantization And Signal Processing.* Boston, MA: Kluwer Academic Publishers

[Geva & Sitte 1991] Geva, S. & Sitte, J.: Adaptive Nearest Neighbour Pattern Classification. *IEEE Transactions on Neural Networks* **2** (1991), 318–322

[Kohonen 1988] Kohonen, T.: *Self-Organization and Associative Memory.* New York, NY: Springer-Verlag

[Murphy & Aha 1994] Murphy, P. & Aha, D.: *UCI repository of machine learning databases – a machine-readable data repository.* Maintained at the Department of Information and Computer Science, University of California, Irvine. Anonymous FTP from `ics.uci.edu` in `/pub/machine-learning-databases`

[Murthy *et al.* 1994] Murthy, S.K., Kasif, S., Salzberg, S.: A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research,* **2**, 1–32

[Quinlan 1992] Quinlan, J.R.: *C4.5 - Programs For Machine Learning.* San Mateo CA: Morgan Kaufmann

[Quinlan 1994] Quinlan, J.R.: Comparing Connectionist & Symbolic Learning Methods. In Hanson, S. J., Drastal, G. A. & Rivest, R. L. (Eds): *Computational Learning Theory and Natural Learning Systems, Vol 1, Constraints and Prospects.* Cambridge MA: MIT Press, 445–456

[Quinlan 1996] Quinlan, J.R.: Improved Use of Continuous Attributes in C4.5. *Journal of Artificial Intelligence Research* **4** 77–90

[Sanchez *et al.* 1998] Sanchez, J.S., Pla, F., Ferri, F.J.: A Voronoi-diagram-based approach to oblique decision tree induction. *Proceedings. Fourteenth International Conference on Pattern Recognition* (1998) 542–544

[Utgoff & Brodley 1991] Utgoff, P.E. & Brodley, C.E. Linear Machine Decision Trees. *Technical Report 10,* University of Massachusetts at Amherst

Making Knowledge Extraction and Reasoning Closer

Fosca Giannotti and Giuseppe Manco

CNUCE - CNR
Via S. Maria 36. 56125 Pisa - Italy
{F.Giannotti,G.Manco}@cnuce.cnr.it

Abstract. The paper shows how a logic-based database language can support the various steps of the KDD process by providing a high degree of expressiveness, and the separation of concerns between the specification level and the mapping to the underlying databases and data mining tools. In particular, the mechanism of user-defined aggregates provided in $\mathcal{LDL}++$ allows to specify data mining tasks and to formalize the mining results in a uniform way. We show how the mechanism applies to the concept of Inductive Databases, proposed in [2,12]. We concentrate on bayesian classification and show how user defined aggregates allow to specify the mining evaluation functions and the returned patterns. The resulting formalism provides a flexible way to customize, tune and reason on both the evaluation functions and the extracted knowledge.

1 Introduction

In recent years, there has been an increasing attention to the problem of formalizing the notion knowledge discovery process. Current knowledge extraction tools and applications, in fact, provide little support to manage the overall process in a uniform way, and to tune the process according to domain knowledge. This is particularly problematic in classification tasks, where the role of domain, or background, knowledge is relevant and may influence the classification results within each step of the KDD process: which attributes discriminate best, how can we characterize a correct/useful profile, which are the useful domain transformations, etc., are all examples of domain dependent notions.

A coherent formalism, capable of dealing uniformly with induced knowledge and background, or domain, knowledge, would represent a significant advance in the design and development of decision support systems, in several challenging application domains. The advantages of such an integrated formalism are, in principle:

- the capability of specifying expert rules, or business rules;
- the ability to tailor a methodology to a specific class of applications.

Other proposals in the current literature have shown that the knowledge discovery process can take great advantage of a powerful knowledge-representation

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 360–371, 2000.
© Springer-Verlag Berlin Heidelberg 2000

and reasoning formalism [15,11,17,5,3,8]. In this context, the notion of *inductive database*, proposed in [2,12], is a first attempt to formalize the notion of interactive mining process. An inductive database provides a unified and transparent view of both inferred (deductive) knowledge, and all the derived patterns, (the induced knowledge) over the data. The user does not care about whether he is dealing with inferred or induced knowledge, and whether the requested knowledge is materialized or not. The only detail he is interested in stems in the high-level specification of the query involving both deductive and inductive knowledge, according to some (either objective or subjective) interestigness quality measure. Interestingly, the notion of Inductive Databases finds a natural generalization in rule-based languages, such as Deductive Databases. In [6], we proposed a model for a such generalization, based on the notion of user-defined aggregate in the logic database $\mathcal{LDL}$++.

In this paper we consider the problem of defining a logic-based knowledge discovery support environment capable of dealing with classification tasks. We extend the approach shown in [6,8] to Naive Bayes Classification, that, to the purpose of this paper, has two main advantages:

1. It is one of the most practical approaches to several types of learning problems.
2. It is particularly simple to represent as an inductive database schema.

As a result, we show how a logic-based database language such as $\mathcal{LDL}$++ [20] can support the various steps of the KDD process by providing: a high degree of expressiveness, the ability to formalize the overall KDD process and the capability of separating the concerns between the specification level and the mapping to the underlying databases and data mining tools.

The paper is organized as follows. In section 2 we introduce the basic features of a Logic Database Language and briefly sketch the mechanism of *user defined aggregates* provided by $\mathcal{LDL}$++. Section 3 describes the generalization of the concept of inductive databases to the case of deductive databases. In section 4 we show how such a mechanism provides a flexible way to custumize, tune and reason on both the evaluation function and the extracted knowledge, by concentrating on the formalization and representation of the bayesian classification [4] data minig task. Finally, a short final section discusses some remarks that the approach issues.

2 Logic Database Languages

Deductive databases are database management systems whose query languages and storage structures are designed around a logical model of data. The underlying technology is an extension to relational databases that increases the power of the query language. We adopt the $\mathcal{LDL}$++ deductive database system, which provides, in addition to the typical deductive features, a highly expressive query language with advanced mechanisms for non-deterministic, non-monotonic and temporal reasoning [7,21].

A remarkable capability of such a language is that of expressing distributive aggregates, which are definable by the user [21]. For example, the following rule illustrates the use of a sum aggregate, which aggregates the values of the relation sales along the dimension Dealer:

$$\texttt{supplierTot(Time, Product, sum}\langle\texttt{Sales}\rangle\texttt{)} \leftarrow \texttt{sales(Time, Product, Store, Sales)}.$$

Such rule corresponds to the SQL statement

```
SELECT Time, Product, SUM(Sales)
FROM sales
GROUP BY Time, Product
```

From a semantic viewpoint, the above rule is a syntactic sugar for a program that exploits the notions of nondeterministic choice and XY-stratification [10,20,7]. In order to compute the following aggregation predicate

$$\texttt{q(Y, aggr}\langle\texttt{X}\rangle\texttt{)} \leftarrow \texttt{p(X, Y)}.$$

we exploit the capability of imposing a nondeterministic order among the tuples of the relation p,

$$\texttt{ordP(Y, nil, nil)} \leftarrow \texttt{p(X, Y)}.$$
$$\texttt{ordP(Z, X, Y)} \leftarrow \texttt{ordP(Z, \_, X), p(Y, Z), choice(X, Y), choice(Y, X)}.$$

Here `nil` is a fresh constant, conveniently used to simplify the program. If the base relation p is formed by k tuples for a given value s of Y, then there are $k!$ possible outcomes for the query `ordP(X, Y)`, namely a set:

$$\{\texttt{ordP(s, nil, nil)}, \texttt{ordP(s, nil, t}_1\texttt{)}, \texttt{ordP(s, t}_1\texttt{, t}_2\texttt{)}, \ldots, \texttt{ordP(s, t}_{k-1}\texttt{, t}_k\texttt{)}\}$$

for each permutation $\{(\texttt{s}, \texttt{t}_1), \ldots, (\texttt{s}, \texttt{t}_k)\}$ of the tuples of P. Therefore, in each possible outcome of the mentioned query, the relation `ordP` is a total (intransitive) ordering of the tuples of p. The double `choice` constraint in the recursive rule specifies that the successor and predecessor of each tuple of p is unique.

As shown in [20], we can then exploit such an ordering to define "recursive" aggregates, i.e., aggregates inductively defined:

$$f(\{x\}) = g(x) \tag{1}$$
$$f(S \cup \{x\}) = h(f(S), x) \tag{2}$$

By defining the base and inductive cases by means of ad-hoc user-defined predicates `single` and `multi`, we can then define an incremental computation of the aggregation function:

$$\texttt{aggrP(Aggr, Z, X, C)} \leftarrow \texttt{ordP(Z, nil, X), X} \neq \texttt{nil, single(Aggr, X, C)}.$$

$$\texttt{aggrP(Aggr, Z, Y, C)} \leftarrow \texttt{ordP(Z, X, Y), aggrP(Aggr, X, C}_1\texttt{), multi(Aggr, Y, C}_1\texttt{, C)}.$$

Finally, the originary rule can be translated into

$$\texttt{q(Y, C)} \leftarrow \texttt{ordP(Y, \_, X)}, \neg\texttt{ordP(Y, X, \_)}, \texttt{aggrP(aggr, Y, X, C)}.$$

Example 1 ([21]). The aggregate sum can be easily defined by means of the following rules:

$$\texttt{single}(\texttt{sum}, \texttt{X}, \texttt{X}).$$

$$\texttt{multi}(\texttt{sum}, \texttt{X}, \texttt{SO}, \texttt{SN}) \leftarrow \texttt{SN} = \texttt{SO} + \texttt{X}.$$

□

In [21], a further extension to the approach is proposed, in order to deal with more complex aggregation functions. Practically, we can manipulate the results of the aggregation function by means of two predicates freturn and ereturn. The rule defininining the aggregation predicate is translated into the following:

$$\texttt{q}(\texttt{Z}, \texttt{R}) \leftarrow \texttt{ordP}(\texttt{Z}, \texttt{X}, \texttt{Y}), \texttt{aggrP}(\texttt{aggr}, \texttt{Z}, \texttt{X}, \texttt{C}), \texttt{ereturn}(\texttt{aggr}, \texttt{Y}, \texttt{C}, \texttt{R}).$$
$$\texttt{q}(\texttt{Z}, \texttt{R}) \leftarrow \texttt{ordP}(\texttt{Z}, \texttt{X}, \texttt{Y}), \neg\texttt{ordP}(\texttt{Z}, \texttt{Y}, \_), \texttt{aggrP}(\texttt{aggr}, \texttt{Z}, \texttt{Y}, \texttt{C}), \texttt{freturn}(\texttt{aggr}, \texttt{C}, \texttt{R}).$$

where the first rule defines *early returns* (i.e., results of intermediate computations), and the second rule defines *final returns*, i.e., results on overall values.

Example 2 ([21]). The aggregate maxpair considers tuples (c_i, n_i), where n_i is a real number, and returns the term c_i with the greater value of n_i. The aggregate can be defined by means of single, multi and freturn:

$$\texttt{single}(\texttt{maxpair}, (\texttt{C}, \texttt{P}), (\texttt{C}, \texttt{P})).$$

$$\texttt{multi}(\texttt{maxpair}, (\texttt{C}, \texttt{P}), (\texttt{C0}, \texttt{P0}), (\texttt{C}, \texttt{P})) \leftarrow \texttt{P} \geq \texttt{P0}.$$
$$\texttt{multi}(\texttt{maxpair}, (\texttt{C}, \texttt{P}), (\texttt{C0}, \texttt{P0}), (\texttt{C0}, \texttt{P0})) \leftarrow \texttt{P} < \texttt{P0}.$$

$$\texttt{freturn}(\texttt{maxpair}, (\texttt{C0}, \texttt{P0}), \texttt{C0}).$$

□

Example 3. Given the relation gate(G,X) specifying the output signal of a gate G, the andGate(Y) predicate should compute the intersection of the signals of all the available gates. The computation can be specified by means of an aggregate and:

$$\texttt{andGate}(\texttt{and}\langle\texttt{X}\rangle) \leftarrow \texttt{gate}(\texttt{G}, \texttt{X}).$$

$$\texttt{single}(\texttt{and}, \texttt{X}, \texttt{X}).$$

$$\texttt{multi}(\texttt{and}, \texttt{X}, \texttt{A}, \texttt{A}) \leftarrow \texttt{X} \neq 0, \texttt{A} \neq 0.$$
$$\texttt{ereturn}(\texttt{and}, 0, \texttt{A}, 0).$$
$$\texttt{freturn}(\texttt{and}, \texttt{X}, \texttt{X}).$$

Notice that the ereturn predicate allows to stop the computation as soon as a 0 gate is found. □

3 Logic-Based Inductive Databases

In [2], an inductive database schema is defined as a pair $\mathcal{R} = (\mathbf{R}, (\mathcal{Q}_\mathbf{R}, e, \mathcal{V}))$, where $\mathbf{R}$ is a database schema, $\mathcal{Q}_\mathbf{R}$ is a collection of patterns, $\mathcal{V}$ is a set of result

values and e is an evaluation function mapping each instance $\mathbf{r}$ of $\mathbf{R}$ and each pattern $\theta \in \mathcal{Q}_{\mathbf{R}}$ in $\mathcal{V}$. An inductive database instance is then defined as a pair $(\mathbf{r}, s)$, where $\mathbf{r}$ is an instance of $\mathbf{R}$ and $s \subseteq \mathcal{Q}_{\mathbf{R}}$.

A typical KDD process operates on both the components of an inductive database, by querying both components of the pair (assuming that s is materialized as a table, and that the value $e(\mathbf{r}, \theta)$ is available for each value θ of s).

A simple yet powerful way of formalizing such ideas in a query language is that of exploiting user-defined aggregates [6]. Practically, we can formalize the inductive part of an inductive database (i.e., the triple $(\mathcal{Q}_{\mathbf{R}}, e, \mathcal{V})$) by means of rules that instantiate the following general schema:

$$s(Z_1, \ldots, Z_k, u\_d\_aggr\langle e_1, \ldots, e_h, X_1, \ldots, X_n\rangle) \leftarrow \mathbf{r}(Y_1, \ldots, Y_m). \qquad (3)$$

Intuitively, this rule defines the format of any subset s of $\mathcal{Q}_{\mathbf{R}}$. $e_1, \ldots, e_h$ specify the components needed to compute the evaluation function e. The patterns in s are obtained from a rearranged subset $X_1, \ldots, X_n$ of the tuples $Y_1, \ldots, Y_m$ in $\mathbf{r}$. The structure of s is defined by the formal specification of the aggregate u_d_aggr, in particular by the `return` rules.

The tuples (θ, v) resulting from the evaluation of such rule, represent patterns in $\mathcal{Q}_{\mathbf{R}}$ and their value in $\mathcal{V}$ according to e. As a result, the "inductive" predicate s itself can be used in the definition of more complex queries.

Example 4. Consider the relation `transaction(Date, Cust, Item, Price, Qty)`. A sample mining scenario for such a table consists in detecting the items in the relation with the average value more than a given threshold. The inductive database has $\mathbf{R} \equiv$ `transaction`, $\mathcal{Q}_{\mathbf{R}} = \{i | i \in dom(\mathbf{R}[Item])\}$ and $e(\mathbf{r}, i) = avg(\{p \times q | (t, i, p, q) \in \mathbf{r}\})$. The above inductive schema is formalized, in accordance to (3) with the following rule:

$$\texttt{s(avgThres}\langle(\sigma, \texttt{Item}, \texttt{Value})\rangle) \leftarrow \texttt{transaction}(\_, \_, \texttt{Item}, \texttt{Price}, \texttt{Qty}), \texttt{Value} = \texttt{Price} \times \texttt{Qty}.$$

Where σ represents the given threshold, and the aggregate `avgThres` is defined, as usual, by means of the predicates

$$\texttt{single(avgThres}, (\texttt{T}, \texttt{I}, \texttt{V}), (\texttt{T}, \texttt{I}, \texttt{V}, 1)).$$

$$\texttt{multi(avgThres}, (\texttt{T}, \texttt{I}, \texttt{VN}), (\texttt{T}, \texttt{I}, \texttt{VO}, \texttt{NO}), (\texttt{T}, \texttt{I}, \texttt{V}, \texttt{N})) \leftarrow \texttt{V} = \texttt{VN} + \texttt{VO}, \texttt{N} = \texttt{NO} + 1.$$
$$\texttt{multi(avgThres}, (\texttt{T}, \texttt{I}, \texttt{VN}), (\texttt{T}, \texttt{I}, \texttt{VO}, \texttt{NO}), (\texttt{T}, \texttt{I}, \texttt{VO}, \texttt{NO})).$$
$$\texttt{multi(avgThres}, (\texttt{T}, \texttt{I}, \texttt{VN}), (\texttt{T}, \texttt{IO}, \texttt{VO}, \texttt{NO}), (\texttt{T}, \texttt{I}, \texttt{VN}, 1)) \leftarrow \texttt{I} \neq \texttt{IO}.$$

$$\texttt{freturn(avgThres}, (\texttt{T}, \texttt{I}, \texttt{V}, \texttt{N}), (\texttt{I}, \texttt{A})) \leftarrow \texttt{A} = \texttt{V}/\texttt{N}, \texttt{A} \geq \texttt{T}.$$

For each item, both the sum and the count of the occurrences is computed[1]. When all the tuples have been considered, the average value of each item is

[1] Here, a naive computation is specified, that potentially computes more than one tuple for each item. Notice, however, that more refined definitions are possible, by exploiting negation and slightly modifying the definition of user defined aggregate.

computed, and returned and as answer if and only if it is greater than the given threshold. □

Notice that the above schema can produce different computations of $\mathbf{s}$ on different clusters of the tuples of $\mathbf{r}$ grouped on the basis of the values of the attributes $Z_1, \ldots, Z_k$. Practically, the results of the above schema, given an instance $\mathbf{r}$ of $\mathbf{R}$, are the tuples $(\sigma_{C_1}(\mathbf{r}), s_1), \ldots, (\sigma_{C_h}(\mathbf{r}), s_h)$, where $C_i \equiv Z_1 = v_{i_1} \wedge \ldots \wedge Z_k = v_{i_k}$ with $v_{i_j} \in dom(Z_j)$, and s_i is the set of patterns related to $\sigma_{C_i}(\mathbf{r})$.

In [6], we showed that the formalism is flexible enough to allow the formalization of association rule mining. In the following, we extend the approach proposed there to the classification data mining task.

4 Bayesian Classification

According to the approach presented in the previous section, we are interested in developing a classification construct based on the notion of user-defined aggregate. We concentrate here on bayesian classification, that is among the most practical approaches to many types of learning problems [13,14]. To summarize, we aim at computing the function

$$\max_c \quad Prob(C = c | A_1 = a_1, \ldots, A_n = a_n)$$

where c is the target attribute and $a_1, \ldots, a_n$ are possible values of the attributes of a relation with schema $\mathbf{R} = \{Z_1, \ldots, Z_k\}$ such that $\{A_1, \ldots, A_n, C\} \subseteq \{Z_1, \ldots, Z_k\}$. By assuming that, for each i, j such that $i \neq j$, A_i and A_j are independent, the above expression is maximized by the same value c that maximizes the expression

$$\max_c \quad Prob(C = c) \cdot \prod_1^n Prob(A_i = a_i | C = c)$$

We can define an inductive database schema as follows. The patterns $\mathcal{Q}_\mathbf{R}$ are represented by the expressions $A_i = a_i \wedge C = c$, where $a_i \in dom(A_i)$ and $c \in dom(C)$. Let $\mathcal{V} = [0, 1]^2$, $\mathbf{r}$ be an instance of $\mathbf{R}$ and $\theta \equiv A_i = a_i \wedge C = c \in \mathcal{Q}_\mathbf{R}$; then $e(\mathbf{r}, \theta) = (Prob(A_i = a_i | C = c), Prob(C = c))$.

By assuming that $Prob(A|B)$ can be estimated as $count(A \wedge B)/count(B)$, we can then define the pair $(Prob(A_i = a_i | C = c), Prob(C = c))$ as a user-defined aggregate[2]. Practically, we define a predicate $\mathbf{s}$ as

$$\mathtt{s}(\mathtt{X}_1, \ldots, \mathtt{X}_m, \mathtt{bayes}(\langle \mathtt{P}, [\mathtt{A}_1, \ldots, \mathtt{A}_n], \mathtt{C} \rangle)) \leftarrow \mathtt{r}(\mathtt{Z}_1, \ldots, \mathtt{Z}_k).$$

where the variables $\mathtt{X}_1, \ldots, \mathtt{X}_m, \mathtt{P}, \mathtt{A}_1, \ldots, \mathtt{A}_n, \mathtt{C}$ are a (possibly rearranged) subset of $\mathtt{Z}_1, \ldots, \mathtt{Z}_k$ and $\mathtt{r}(\mathtt{Z}_1, \ldots, \mathtt{Z}_k)$ is either an extensional or an intensional predicate.

[2] Notice that such an approach fails to deal with 0 probabilities. However, simple corrections can be done, as suggested, e.g., in [14, chapter 6]. For example, we can weight the proposed estimate of $Prob(A|B)$ with prior uniform probabilities.

The result of such an evaluation is the set of conditional probabilities of each of the possible values of $\mathtt{A_i}$, given any possible value of C, and a weight P associated to the tuple $\mathtt{A_1}, \ldots, \mathtt{A_n}$.

We can define, as usual, the `single`, `multi` and `freturn` predicates:

$$\mathtt{single}(\mathtt{bayes}, (\mathtt{P}, \mathtt{F}, \mathtt{C}), (\mathtt{P}, \mathtt{Fs})) \leftarrow \mathtt{init}(\mathtt{F}, \mathtt{C}, \mathtt{P}, \mathtt{Fs}).$$

$$\mathtt{multi}(\mathtt{bayes}, (\mathtt{P}, \mathtt{F}, \mathtt{C}), (\mathtt{NO}, \mathtt{FO}), (\mathtt{NO} + \mathtt{P}, \mathtt{FN})) \leftarrow \mathtt{update}(\mathtt{F}, \mathtt{C}, \mathtt{P}, \mathtt{FO}, \mathtt{FN}).$$

The tuple (P, C, F) represents a tuple in the database. More precisely, C is the target attribute, F is the collection of the relevant features and P is the weight associated to the current tuple. The `init` predicate builds a list of tuples $(f_i, c_i, n_{f_i}, n_{c_i})$, representing respectively the feature f_i, the target attribute associated to c_i, the (current) frequency n_{f_i} of f_i, c_i and the related frequency n_{c_i} of c_i. The `update` predicate updates the list FO with the tuples available from C and F. For each quadruple $(f_i, c_i, n_{f_i}, n_{c_i})$ in FO and for each pair (f, c) where $f \in F$, n_{c_i} is incremented if $c = c_i$ and n_{f_i} is incremented if both $f = f_i$ and $c = c_i$.

As a final result, the collected tuples allow the computation of conditional probabilities:

$$\mathtt{freturn}(\mathtt{bayes}, (\mathtt{N}, \mathtt{FO}), (\mathtt{C}, \mathtt{F}, \mathtt{P_C}, \mathtt{P_{F,C}})) \leftarrow \mathtt{member}((\mathtt{F}, \mathtt{C}, \mathtt{C_F}, \mathtt{C_C}), \mathtt{FO}), \mathtt{P_{F,C}} = \mathtt{C_F}/\mathtt{C_C}, \mathtt{P_C} = \mathtt{C_C}/\mathtt{N}.$$

Let us consider the extensional predicate:

$$\mathtt{playTennis}(\mathtt{Outlook}, \mathtt{Temperature}, \mathtt{Humidity}, \mathtt{Wind}, \mathtt{Play})$$

with extension in table 1. A simple classifier on such a relation is built by means of the rule

$$\mathtt{classifier}(\mathtt{bayes}\langle(1, [\mathtt{Outlook}, \mathtt{Temp}, \mathtt{Humidity}, \mathtt{Wind}], \mathtt{Play})\rangle) \leftarrow \mathtt{playTennis}(\mathtt{Outlook}, \mathtt{Temp}, \mathtt{Humidity}, \mathtt{Wind}, \mathtt{Play}).$$

Example 5. A query $\mathtt{classifier}(\mathtt{C}, \mathtt{F}, \mathtt{P_C}, \mathtt{P_F})$ against such a database returns the answers

(no, sunny, 0.357143, 0.6)	(no, hot, 0.357143, 0.4)
(no, weak, 0.357143, 0.4)	(no, strong, 0.357143, 0.6)
(yes, hot, 0.642857, 0.222222)	(yes, high, 0.642857, 0.333333)
(yes, rain, 0.642857, 0.444444)	(yes, mild, 0.642857, 0.444444)
(yes, normal, 0.642857, 0.666667)	(yes, strong, 0.642857, 0.333333
(no, cool, 0.357143, 0.2)	(no, normal, 0.357143, 0.2)
(yes, sunny, 0.642857, 0.222222)	(no, rain, 0.357143, 0.2)
(no, high, 0.357143, 0.8)	(yes, overcast, 0.642857, 0.333333)
(yes, weak, 0.642857, 0.666667)	(yes, cool, 0.642857, 0.333333))
(no, overcast, 0.357143, 0.2)	(no, mild, 0.357143, 0.4)

Such tuples represent a classification model that can be used to classify any new tuple. □

Table 1. Sample `playTennis` facts.

```
playTennis(overcast, hot, normal, weak, yes)
playTennis(sunny, hot, high, strong, no)
playTennis(overcast, cool, normal, strong, no)
playTennis(rain, cool, normal, strong, yes)
playTennis(sunny, mild, high, weak, no)
playTennis(rain, mild, normal, weak, yes)
playTennis(overcast, mild, high, strong, yes)
playTennis(sunny, hot, high, weak, no)
playTennis(overcast, hot, high, weak, yes)
playTennis(rain, cool, normal, weak, yes)
playTennis(rain, mild, high, weak, yes)
playTennis(sunny, cool, normal, weak, yes)
playTennis(sunny, mild, normal, strong, yes)
playTennis(rain, mild, high, strong, no)
```

Once the classification model is trained, the classifier can be easily built:

$$\begin{aligned}\texttt{classify}(\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{maxpair}\langle(\texttt{P},\texttt{Prob})\rangle) \leftarrow\ & \texttt{classifier}(\texttt{P},\texttt{O},\texttt{Prob}_\texttt{o},\texttt{Prob}_\texttt{p}),\\ & \texttt{classifier}(\texttt{P},\texttt{T},\texttt{Prob}_\texttt{t},\texttt{Prob}_\texttt{p}),\texttt{classifier}(\texttt{P},\texttt{H},\texttt{Prob}_\texttt{h},\texttt{Prob}_\texttt{p}),\\ & \texttt{classifier}(\texttt{P},\texttt{W},\texttt{Prob}_\texttt{w},\texttt{Prob}_\texttt{p}),\\ & \texttt{Prob} = \texttt{Prob}_\texttt{p} \times \texttt{Prob}_\texttt{o} \times \texttt{Prob}_\texttt{t} \times \texttt{Prob}_\texttt{h} \times \texttt{Prob}_\texttt{w}\end{aligned}$$

where the `maxpair` aggregate is defined in example 2. The above defined predicate $\texttt{classify}(\texttt{F}_1,\ldots,\texttt{F}_\texttt{n},\texttt{Target})$ guesses the value of `Target` on the basis of the values of the interesting features $\texttt{F}_1,\ldots,\texttt{F}_\texttt{n}$, according to the classification model defined by the relation `classifier`.

Notice that, in general, the definition of the predicate `classify` depends from the model defined by the predicate `classifier`.

$$\begin{aligned}\texttt{classify}(\texttt{X}_1,\ldots,\texttt{X}_\texttt{m},\texttt{F}_1,\ldots,\texttt{F}_\texttt{n},\texttt{maxpair}\langle(\texttt{C},\texttt{P})\rangle) \leftarrow\ & \\ & \texttt{classifier}(\texttt{X}_1,\ldots,\texttt{X}_\texttt{m},\texttt{C},\texttt{F}_1,\texttt{P}_\texttt{C},\texttt{P}_{\texttt{F}_1}),\\ & \quad \texttt{classifier}(\texttt{X}_1,\ldots,\texttt{X}_\texttt{m},\texttt{C},\texttt{F}_2,\texttt{P}_\texttt{C},\texttt{P}_{\texttt{F}_2}),\\ & \qquad \vdots\\ & \quad \texttt{classifier}(\texttt{X}_1,\ldots,\texttt{X}_\texttt{m},\texttt{C},\texttt{F}_\texttt{n},\texttt{P}_\texttt{C},\texttt{P}_{\texttt{F}_\texttt{n}}),\\ & \qquad \texttt{P} = \texttt{P}_{\texttt{F}_1} \times \ldots \times \texttt{P}_{\texttt{F}_\texttt{n}}.\end{aligned}$$

In the following we shall concentrate only in the definition of the `classifier` predicate, and shall omit the definition of `classify`. The rest of the section shows some examples of complex queries whithin the resulting logic language. We shall refer to table 1 as a running example.

It is easy to build a predicate that evaluates the goodness (w.r.t. some quality measure) of the classifier built in the previous section.

Example 6. The high-level specification task *"Compute the misclassification rate of the trained model"* is formalized by the following rule:

$$\begin{aligned}\texttt{misclassified}(\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{Play},\texttt{Predicted}) \leftarrow\ & \texttt{testSet}(\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{Play}),\\ & \texttt{classify}(\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{Predicted}),\texttt{Play} \neq \texttt{Predicted}.\end{aligned}$$

When run against the `playTennis` table, the `misclassified` relation returns the tuples

(`overcast`, `cool`, `normal`, `strong`, `no`, `yes`)
(`overcast`, `mild`, `high`, `strong`, `yes`, `no`)
(`rain`, `mild`, `high`, `strong`, `no`, `yes`)

representing the misclassified portion of the table. □

The predicate testSet can be defined either as an extensional predicate or as an intensional predicate. This suggests a way for directly defining a boosting technique.

Example 7 (Boosting). The high-level specification task for boosting *"compute an increasing sequence of classifiers such that each classifier is built by increasing the weight of the tuples misclassified by the preceding classifier"* can be defined by the following set of rules:

$$\texttt{case}(0,1,0,T,H,W,P) \leftarrow \texttt{playTennis}(0,T,H,W,P).$$
$$\texttt{case}(I+1,WW+1,0,T,H,W,P) \leftarrow \texttt{case}(I,WW,0,T,H,W,P), \texttt{misclassified}(I,0,T,H,W,P).$$
$$\texttt{case}(I+1,WW,0,T,H,W,P) \leftarrow \texttt{case}(I,WW,0,Temp,H,W,P), \neg\texttt{misclassified}(I,0,T,H,W,\_).$$

$$\texttt{classifier}(I,\texttt{bayes}\langle(WW,[0,T,H,W],P)\rangle) \leftarrow \texttt{case}(I,WW,0,T,H,W,P).$$
$$\texttt{misclassified}(I,0,T,H,W,P) \leftarrow \texttt{case}(I,WW,0,T,H,W,P), \texttt{classify}(I,0,T,H,W,Pred), P \neq Pred.$$

Classifiers are incrementally built and identified by the stage argument I. At each stage, the training-set is built by incrementing the weight of the misclassified tuples. In order to obtain the misclassification rate of each classifier built with the above technique, we have to count the misclassification rate of each classifier:

$$\texttt{totMisclassified}(I,\texttt{count}\langle(0,T,H,W,P)\rangle) \leftarrow \texttt{miclassified}(I,0,T,H,W,P).$$

The first argument of the totMisclassied predicate represents the classifier ID, and the second argument represents the number of tuples misclassified by the classifier. By considering the first 10 classifiers built we obtain the following misclassification rate for each classifier:

totMisclassified(6, 1) totMisclassified(9, 2) totMisclassified(4, 3)
totMisclassified(1, 3) totMisclassified(3, 2) totMisclassified(5, 2)
totMisclassified(8, 3) totMisclassified(0, 3) totMisclassified(2, 3)
totMisclassified(7, 5)

□

Example 8 (Meta-Learning). Auto-focusing mechanisms can be easily defined. We can *"train a classifier as a coordinator of a set of classifiers built by boosting, where each classifier has a votation weight depending by its misclassification rate"*:

$$\texttt{votation}(0,T,H,W,C,\texttt{sum}\langle V\rangle) \leftarrow \texttt{classify}(I,0,T,H,W,C,P), \texttt{totMisclassified}(I,T), V = 1/T.$$

$$\texttt{boostClassify}(0,T,H,W,\texttt{maxpair}\langle(C,N)\rangle) \leftarrow \texttt{votation}(0,T,H,W,C,N).$$

The misclassification rate of the boosting classifier is computed by the rule

$$\texttt{boostMisclassified}(\texttt{count}\langle(0,T,H,W,C)\rangle) \leftarrow \texttt{playTennis}(0,T,H,W,C), \texttt{boostClassify}(0,T,H,W,Pred), C \neq Pred.$$

□

Example 9 (Cross-Validation). In the following we assume that each row of the `playTennis` has an identifier `I`. A cross-validation technique can be used to evaluate the prediction accuracy of the classification task against a given dataset. Practically, we can randomly split a dataset into `K` different training sets and test-sets, and then compute the average prediction accuracy of the adopted classification method.

$$\texttt{crossValidate}(\texttt{K},\texttt{N},\texttt{bayes}\langle(1,[\texttt{O},\texttt{T},\texttt{H},\texttt{W}],\texttt{P})\rangle) \leftarrow \texttt{nthTrainingSet}(\texttt{K},\texttt{N},\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{P}).$$

$$\texttt{nthTrainingSet}(\texttt{K},\texttt{N},\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{P}) \leftarrow \texttt{playTennis}(\texttt{I},\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{P}), \neg\texttt{belongs}(\texttt{I},\texttt{K},\texttt{N}).$$

$$\texttt{nthTestSet}(\texttt{K},\texttt{N},\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{P}) \leftarrow \texttt{playTennis}(\texttt{I},\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{P}), \texttt{belongs}(\texttt{I},\texttt{K},\texttt{N}).$$

$$\texttt{countMisclassified}(\texttt{K},\texttt{N},\texttt{count}\langle(\texttt{O},\texttt{T},\texttt{H},\texttt{W})\rangle) \leftarrow \texttt{nthTestSet}(\texttt{K},\texttt{N},\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{P}), \texttt{classify}(\texttt{K},\texttt{N},\texttt{O},\texttt{T},\texttt{H},\texttt{W},\texttt{Pred}), \texttt{P} \neq \texttt{Pred}.$$

$$\texttt{crossValidation}(\texttt{K},\texttt{avg}\langle\texttt{C}\rangle) \leftarrow \texttt{countMisclassified}(\texttt{K},\texttt{N},\texttt{C}).$$

Here, the predicate `belongs(I,K,N)` specifies the splitting policy of the dataset[3]. □

5 Final Remark

A very desirable property of systems for data mining and knowledge discovery is the capability of separating the concerns between the conceptual/logical design and the phisical implementation of data mining applications. To this purpose, the approach described in this paper and in [6] proposes a uniform declarative specification of the various steps of the knowledge discovery process.

In this paper our primary aim was the investigation of the integration of a deductive query language with a classification engine. We have shown that the formalism of user-defined aggregates is powerful enough to (1) model the notion of inductive database, and (2) to specify flexible query answering capabilities. The main difference with other database-oriented approaches is that the capability of defining logical expressions has very desirable characteristics in order to construct the knowledge processing engine.

The paper was not concerned with efficiency issues. As a matter of fact, the problem of efficiently coupling data mining with database systems is common in many approaches currently existing in the literature. It has been experimentally shown [1,16] that specialized algorithms (provided with specialized data structures) have a better performance than database-oriented approaches. Hence, in

[3] For example, any tuple can be randomly assigned to any value n between 0 and n. In our experiment, we implemented the simple policy of specifying the test set n as composed by all the tuples i such that $mod(i,k) = n$.

order to improve performance considerably, a thorough modification of the underlying database abstract machine should be investigated. Notice in fact that, with respect to ad hoc algorithms, when the programs specified in the previous sections are executed on a logic abstract machine, the only available optimizations for such programs are the traditional deductive databases optimizations [7]. Such optimizations techniques, however, can be (and sometimes need to be) further improved by adding ad-hoc optimizations.

To the purpose of this paper, it can be assumed to accept a reasonable worsening in performance, by describing the aggregation formalism as a semantically clean representation formalism, and demanding the computational effort to external ad-hoc inductive engines, in the style of [8,9]. This, however, is only a partial solution to the problem, in that more refined optimization techniques can be adopted, as envisaged in [6]. Some interesting steps in this direction have been made: e.g., [18] proposes an approach to the optimization of datalog aggregation-based queries.

Another interesting way of coping with efficiency is that of identifying a set of relevant features that can be transferred into more specialized and efficient languages. As an example, [19] study how to provide relational database systems with the mechanism of user-defined aggregates. This suggests that specialized languages for mining/olap tasks could benefit of even a subset of the features of a logic database language, easy to implement in an efficient way (such as, for example, the mechanism of rules for describing the process, or standard input/output interfaces for the interaction between mining and querying). However, a more detailed discussion of such problems is postponed to future work.

References

1. R. Agrawal, S. Sarawagi, and S. Thomas. Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications. In *Procs. of ACM-SIGMOD'98*, 1998.
2. J-F. Boulicaut, M. Klemettinen, and H. Mannila. Querying Inductive Databases: A Case Study on the MINE RULE Operator. In *Procs. 2nd European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD98)*, volume 1510 of *Lecture Notes in Computer Science*, pages 194–202, 1998.
3. J-F. Boulicaut, P. Marcel, and C. Rigotti. Query Driven Knowledge Discovery in Multidimensional Data. In *Procs. of the ACM international workshop on Data warehousing and OLAP*, pages 87–93, 1999.
4. C. Elkan. Boosting and Naive Bayesian Learning. In *Procs. of the International Conference on Knowledge Discovery and Data Mining (KDD-97)*, 1997.
5. U.M. Fayyad, G. Piatesky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/the MIT Press, 1996.
6. F. Giannotti and G. Manco. Querying Inductive Databases via Logic-Based User Defined Aggregates. In *Procs. of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, number 1704 in Lecture Notes in Artificial Intelligence, pages 125–135, September 1999.
7. F. Giannotti, G. Manco, M. Nanni, and D. Pedreschi. Nondeterministic, Nonmonotonic Logic Databases. *IEEE Trans. on Knowledge and Data Engineering*, 2000. To appear.

8. F. Giannotti, G. Manco, M. Nanni, D. Pedreschi, and F. Turini. Integration of deduction and induction for mining supermarket sales data. In *Proceedings of the International Conference on Practical Applications of Knowledge Discovery (PADD99)*, April 1999.
9. F. Giannotti, G. Manco, M. Nanni, D. Pedreschi, and F. Turini. Using Deduction for Intelligent Data Analysis. Technical Report B4-1999-02, CNUCE Institute of CNR, January 1999. Submitted for publication.
10. F. Giannotti, D. Pedreschi, and C. Zaniolo. Semantics and Expressive Power of Non Deterministic Constructs for Deductive Databases. In *Journal of Logic Programming*, 1999.
11. J. Han. Towards On-Line Analytical Mining in Large Databases. *Sigmod Records*, 27(1):97–107, 1998.
12. H. Mannila. Inductive databases and condensed representations for data mining. In *International Logic Programming Symposium*, pages 21–30, 1997.
13. D. Michie, D.J. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, New York, 1994.
14. J. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
15. S. Ceri R. Meo, G. Psaila. A New SQL-Like Operator for Mining Association Rules. In *Proceedings of The Conference on Very Large Databases*, pages 122–133, 1996.
16. S. Ruggieri. Efficient C4.5. Technical report, Department of Computer Science, University of Pisa, January 2000. Available at `http://www-kdd.di.unipi.it`.
17. W. Shen, K. Ong, B. Mitbander, and C. Zaniolo. Metaqueries for Data Mining. In *Advances in Knowledge Discovery and Data Mining*, pages 375–398. AAAI Press/The MIT Press, 1996.
18. D. Tsur et al. Query Flocks: A Generalization of Association-Rule Mining. In *Proc. ACM Conf. on Management of Data (Sigmod98)*, pages 1–12, 1998.
19. H. Wang and C. Zaniolo. User defined aggregates in database languages. In *Seventh International Workshop on Database Programming Languages*, september 1999.
20. C. Zaniolo, N. Arni, and K. Ong. Negation and Aggregates in Recursive Rules: The $\mathcal{LDL}$++ Approach. In *Proc. 3rd Int. Conf. on Deductive and Object-Oriented Databases (DOOD93)*, volume 760 of *Lecture Notes in Computer Science*, 1993.
21. C. Zaniolo and H. Wang. Logic-Based User-Defined Aggregates for the Next Generation of Database Systems. *The Logic Programming Paradigm: Current Trends and Future Directions.* Springer Verlag, 1998.

Discovery of Relevant Weights by Minimizing Cross-Validation Error

Kazumi Saito[1] and Ryohei Nakano[2]

[1] NTT Communication Science Laboratories
2-4 Hikaridai, Seika, Soraku, Kyoto 619-0237 Japan
saito@cslab.kecl.ntt.co.jp
[2] Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466-8555 Japan
nakano@ics.nitech.ac.jp

Abstract. In order to discover relevant weights of neural networks, this paper proposes a novel method to learn a distinct squared penalty factor for each weight as a minimization problem over the cross-validation error. Experiments showed that the proposed method works well in discovering a polynomial-type law even from data containing irrelevant variables and a small amount of noise.

1 Introduction

Neural networks can be utilized as a core technique in some KDD (Knowledge Discovery and Data mining) applications such as scientific discovery [2,1]. One important research subject of neural networks is to improve the generalization performance. Here the generalization means the performance on new data. It is widely known that adding some penalty term to a standard training error term can lead to significant improvements in network generalization. As for squared penalty, a single penalty factor is often conveniently used. If we can develop a method that automatically adjusts a distinct penalty factor for each weight, several advantages can be expected, i.e., the generalization performance will be still more improved; the readability of discovered laws will be improved; such a squared penalty term is consistent with any linear scaling of variables; and suitable penalty factors can be determined without inaccurate estimation.

2 Optimal Penalty Factor Calculation

Let $(x_1, \cdots, x_K, y)$ be a vector of variables describing each example, where x_k is a numeric or nominal explanatory variable and y is a numeric target variable. Here we assume that each nominal explanatory variable is described as a *dummy variable*. As a class of numeric formula $y(\boldsymbol{x}; \boldsymbol{\Theta})$, we consider a generalized polynomial expressed by

$$y(\boldsymbol{x}; \boldsymbol{\Theta}) = w_0 + \sum_{j=1}^{J} w_j \prod_{k=1}^{K} x_k^{w_{jk}} = w_0 + \sum_{j=1}^{J} w_j \exp\left(\sum_{k=1}^{K} w_{jk} \ln x_k\right), \quad (1)$$

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 372–375, 2000.
© Springer-Verlag Berlin Heidelberg 2000

where each parameter w_j or w_{jk} is an unknown real number, and J is an unknown integer corresponding to the number of terms. $\boldsymbol{\Theta}$ is an M-dimensional parameter vector constructed by arranging parameters $w_j, j = 0, \cdots, J$, and $w_{jk}, j = 1, \cdots, J, k = 1, \cdots, K$. Let $D = \{(\boldsymbol{x}^\mu, y^\mu), \mu = 1, \cdots, N\}$ be a set of training examples, where N is the number of examples. Here we assume that each training example $(\boldsymbol{x}^\mu, y^\mu)$ is independent and identically distributed. Now, our ultimate goal of the law discovery is defined as a problem of minimizing the generalization error, that is, to find the optimal estimator $\boldsymbol{\Theta}^*$ that minimizes

$$\mathcal{G}(\boldsymbol{\Theta}^*) = E_D E_T \left(y^\nu - y(\boldsymbol{x}^\nu; \boldsymbol{\Theta}^*(D))\right)^2, \tag{2}$$

where $T = (\boldsymbol{x}^\nu, y^\nu)$ denotes test data independent of the training data D. The *least-squares estimate* of $\boldsymbol{\Theta}^*$, denoted by $\widehat{\boldsymbol{\Theta}}$, minimizes the error sum of squares

$$\mathcal{E}_1(\boldsymbol{\Theta}) = \frac{1}{2} \sum_{\mu=1}^{N} \left(y^\mu - y(\boldsymbol{x}^\mu; \boldsymbol{\Theta})\right)^2. \tag{3}$$

However, this estimation is likely to over-fit to the training data; thus, we cannot usually obtain good results in terms of the generalization performance.

As we have already mentioned, it is widely known that adding some penalty term to Eq. (3) can lead to significant improvements in network generalization. Here a simple penalized target function using a single factor is given as below.

$$\mathcal{E}_2(\boldsymbol{\Theta}) = \mathcal{E}_1(\boldsymbol{\Theta}) + \frac{1}{2} \exp(\lambda) \sum_{m=1}^{M} \theta_m^2, \tag{4}$$

where $\exp(\lambda)$ is a penalty factor and $\theta_m \in \boldsymbol{\Theta}$. Here since the penalty factor must be non-negative, we adopted $\exp(\lambda)$, instead of a standard parameterization λ.

To improve both the generalization performance and the readability, we consider a distinct penalty factor for each weight. Let $\boldsymbol{\lambda}$ be an M-dimensional vector $(\lambda_1, \cdots, \lambda_M)^T$, and $\boldsymbol{\Lambda}$ be an M-dimensional diagonal matrix whose diagonal elements are defined by $\Lambda_{mm} = \exp(\lambda_m)$ for $m = 1, \cdots, M$, where $\boldsymbol{a}^T$ denotes a transposed vector of $\boldsymbol{a}$. Then, the discovery of laws subject to Eq. (1) can be defined as the following learning problem in neural networks. That is, the problem is to find the $\boldsymbol{\Theta}$ that minimizes the following objective function for weights

$$\mathcal{E}(\boldsymbol{\Theta}) = \mathcal{E}_1(\boldsymbol{\Theta}) + \frac{1}{2} \boldsymbol{\Theta}^T \boldsymbol{\Lambda} \boldsymbol{\Theta}. \tag{5}$$

Now, we introduce an objective function for penalty factors derived from the procedure of *cross-validation*, and propose *MCV (Minimum Cross-Validation) regularizer*. The procedure of cross-validation divides the data D at random into S distinct segments $(G_s,\ s = 1, \cdots, S)$, and uses $S-1$ segments for training, and uses the remaining one for the test. This process is repeated S times by changing the remaining segment, and the generalization performance is evaluated by using the following MSE (mean squared error) over all S test results.

$$MSE_{CV} = \frac{1}{N} \sum_{s=1}^{S} \sum_{\nu \in G_s} \left(y^\nu - y(\boldsymbol{x}^\nu; \widehat{\boldsymbol{\Theta}}_s)\right)^2. \tag{6}$$

Here $\widehat{\boldsymbol{\Theta}}_s$ denotes the optimal weights obtained by minimizing the following objective function for weights

$$\mathcal{E}_s(\boldsymbol{\Theta}_s) = \frac{1}{2} \sum_{\mu \notin G_s} (y^\mu - y(\boldsymbol{x}^\mu; \boldsymbol{\Theta}_s))^2 + \frac{1}{2} \boldsymbol{\Theta}_s^T \boldsymbol{\Lambda} \boldsymbol{\Theta}_s. \tag{7}$$

The extreme case of $S = N$ is known as the *leave-one-out* method, which is often used for a small size of data. Note that Eq. (6) is regarded as a reasonable approximation to Eq. (2) for a given data set D. According to the *implicit function theorem*, since $\widehat{\boldsymbol{\Theta}}_s$ can be regarded as a vector consisting of implicit functions of $\boldsymbol{\lambda}$, Eq. (6) can be defined as the objective function for penalty factors. Thus, we can calculate $\widehat{\boldsymbol{\lambda}}$ which minimizes Eq. (6). Then, by using $\widehat{\boldsymbol{\lambda}}$, we can calculate $\widehat{\boldsymbol{\Theta}}$ which minimizes Eq. (5). Finally, $\widehat{\boldsymbol{\Theta}}$ is adopted as the final weight vector of the discovered law.

3 Evaluation by Experiments

We consider an artificial law (function) described by

$$y = 2 + 3x_1^{+1} x_2^{-0.02} + 4x_3^{-1} x_4^{+0.02} \tag{8}$$

where we have 9 numeric explanatory variables. Clearly, variables $x_5, \cdots, x_9$ are irrelevant to Eq. (8). Each example is generated as follows: each value of numeric variables $x_1, \cdots, x_9$ is randomly generated in the range of $(0, 1)$, and we get the corresponding value of y by calculating Eq. (8) and adding Gaussian noise with a mean of 0 and a standard deviation of 0.1. The number of examples is set to 200 ($N = 200$). Before the analysis, the following scaling was applied to the variables: $\widetilde{y} = (y - mean(y))/std(y)$, and $\ln \widetilde{x}_k = \ln x_k - mean(\ln x_k)$, $k = 1, \cdots, 9$.

In the experiments, the initial values for the weights w_{jk} were independently generated according to a normal distribution with a mean of 0 and a standard deviation of 1; the initial values for the weights w_j were set to 0. The initial values for the penalty factors $\boldsymbol{\lambda}$ were set to $\mathbf{0}$, i.e., $\boldsymbol{\Lambda}$ was set to the identical matrix. The iteration was terminated when the gradient vector was sufficiently small, i.e., $\max_m\{\|\partial/\partial\theta_m \, \mathcal{E}(\boldsymbol{\Theta})\|\} < 10^{-6}$ for learning over $\boldsymbol{\Theta}$; $\max_m\{\|\partial/\partial\lambda_m \, MSE_{CV}(\boldsymbol{\lambda})\|\} < 10^{-6}$ for learning over $\boldsymbol{\lambda}$.

MCV regularizer was compared with two conventional methods, no-penalty method and single-factor method, where the objective functions of these conventional methods are Eq. (3) and Eq. (4), respectively. Figure 1(a) shows the learning results of these three methods, where the RMSE (root mean squared error) was used for evaluation; the number of hidden units J was fixed at the correct number 2; the cross-validation error was calculated by using the leave-one-out method, i.e., $S = N$; and the generalization performance was measured by using a set of noise-free 10,000 test examples generated independently to the training examples. This figure shows that the RMSE for the training data was almost the same for each method; both the RMSE for the cross-validation and the RMSE for the test data were clearly decreased by using MCV regularizer;

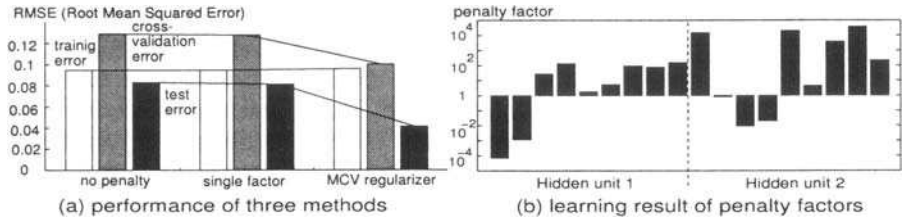


Fig. 1. Experimental results of artificial data

and the performance of the single factor method was almost comparable to those of the no penalty method.

An example of the laws discovered by the no penalty method was as follows:

$$y = 2.0306$$
$$+2.9791x_1^{+1.0024}x_2^{-0.0203}x_3^{+0.0035}x_4^{-0.0029}x_5^{+0.0073}x_6^{+0.0056}x_7^{+0.0010}x_8^{+0.0022}x_9^{-0.0036}$$
$$+3.9993x_1^{+0.0008}x_2^{+0.0004}x_3^{-1.0003}x_4^{+0.0201}x_5^{-0.0005}x_6^{-0.0011}x_7^{-0.0002}x_8^{-0.0002}x_9^{+0.0011}$$

where the weight values were rounded off to the fourth decimal place. Note that these weight values were transformed so as to correspond to the original scale of variables. Although a law almost equivalent to the true one was found, it is difficult to select only the relevant weights from this result. While an example of the laws discovered by MCV regularizer was as follows:

$$y = 2.0118$$
$$+2.9792x_1^{+0.9941}x_2^{-0.0190}x_3^{-0.0000}x_4^{-0.0000}x_5^{+0.0019}x_6^{+0.0007}x_7^{+0.0000}x_8^{+0.0000}x_9^{+0.0000}$$
$$+3.9987x_1^{+0.0000}x_2^{+0.0001}x_3^{-0.9999}x_4^{+0.0197}x_5^{-0.0000}x_6^{-0.0006}x_7^{-0.0000}x_8^{-0.0000}x_9^{+0.0003}$$

Clearly, the irrelevant weight values were greatly suppressed.

Figure 1(b) shows the learning result of the penalty factors. This figure indicates that only the penalty factors for the relevant weights became small enough, i.e., we can easily select only the relevant weights. Therefore, it was shown that the MCV regularizer simultaneously improves the generalization performance and readability, without care of variable scaling and a candidate determination for the penalty factors.

References

1. R. Nakano and K. Saito. Discovery of a set of nominally conditioned polynomials. In *Proc. 2nd Int. Conf. on Discovery Science, LNAI 1721*, pages 287–298, 1999.
2. K. Saito and R. Nakano. Law discovery using neural networks. In *Proc. 15th Int. Joint Conf. on Artificial Intelligence*, pages 1078–1083, 1997.

Efficient and Comprehensible Local Regression

Luís Torgo

LIACC-FEP, University of Porto
R. Campo Alegre, 823 – 4150 Porto – Portugal
ltorgo@ncc.up.pt http://www.ncc.up.pt/~ltorgo

Abstract. This paper describes an approach to multivariate regression that aims at improving the computational efficiency and comprehensibility of local regression techniques. Local regression modeling is known for its ability to accurately approximate quite diverse regression surfaces with high accuracy. However, these methods are also known for being computationally demanding and for not providing any comprehensible model of the data. These two characteristics can be regarded as major drawbacks in the context of a typical data mining scenario. The method we describe tackles these problems by integrating local regression within a partition-based induction method.

1 Introduction

This paper describes a hybrid approach to multivariate regression problems. Multivariate regression is a well known data analysis problem that can be loosely defined as the study of the relationship between a target continuous variable and a set of other input variables based on a sample of cases. In many important regression domains we cannot assume any particular functional form for the model describing this relationship. This type of problems demand for what is usually known as non-parametric approaches. An example of such techniques is local regression modeling (*e.g.* [3]). The basic idea behind local regression consists of delaying the task of obtaining a model till prediction time. Instead of fitting a single model to all given data these methods obtain one model for each query case using only the *most similar* training cases. As a result of this methodology these techniques do not produce any visible and comprehensible model of the given training data. Moreover, for each query point its "neighbors" have to be found, which is a time-consuming task for any reasonably large problem. Still, these models are able to easily adapt to any form of regression surface, which leads to large advantages in terms of their ability to approximate a wide range of functions. In this paper we address the drawbacks of local models by integrating them with regression trees.

2 Local Regression Modeling

According to Cleveland and Loader [3] local regression modeling traces back to the 19th century. These authors provide a historical survey of the work done since then. In

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 376-379, 2000.
© Springer-Verlag Berlin Heidelberg 2000

this paper we focus on one particular type of local modeling, namely kernel regression. Still, the described methodology is applicable to other local models. Within kernel regression a prediction for a query case is obtained by an averaging process over the most similar training cases. The central issue of these models is thus the notion of similarity, which is determined using a particular metric over the multidimensional space defined by the input variables. Given a data set $\{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{n}$, where $\mathbf{x}_i$ is a vector of input variable values, a kernel model prediction for a query case $\mathbf{x}_q$ is obtained by,

$$k(\mathbf{x}_q) = \frac{1}{SKs} \sum_{i=1}^{n} K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \times y_i \tag{1}$$

where,

$d(.)$ is the distance function between two instances;
$K(.)$ is a kernel (weighing) function;
h is a bandwidth (or neighbourhood size) value;
and SKs is the sum of all weights, *i.e.* $SKs = \sum_{i=1}^{n} K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right)$.

In this work we have used an Euclidean distance function together with a gaussian kernel (see [1] for an overview of these and other alternatives).

A kernel prediction can be seen as a weighed average of the target variable values of the training cases that are nearer to the query point. Each of the training cases within a specified distance (the bandwidth h) enter this averaging. Their weight is inversely proportional to the distance to the query, according to the $K(.)$ gaussian function.

The classical definition of the knowledge discovery in databases [4] refers this process as striving to identify valid, novel, potentially useful, and ultimately *understandable* patterns in data. From the perspective of understandability the local regression framework described above is very poor. Another characteristic of a typical data mining problem is its high dimensionality, *i.e.* the large number of cases and/or variables. Local modeling has a very high computational complexity if applied as described above. In effect, the prediction for each query case demands a look-up over all training cases to search for the most similar instances. This process has a complexity of the order of $O(n \times v)$ for each test case, where n is the number of training cases, and v is the number of variables.

3 Local Regression Trees

Regression trees (*e.g.* [2]) are non-parametric models that have as main advantages a high computational efficiency and a good compromise between comprehensibility and predictive accuracy. A regression tree can be seen as a partitioning of the input space.

This partitioning is described by a hierarchy of logical tests on the input variables. Standard regression trees usually assume a constant target variable value within each partition.

The regression method we propose consists of using local regression in the context of the partitions defined by a regression tree. The resulting model differs from a regression tree only in prediction tasks. Given a query case we drop it down the tree until a leaf is reached, as in standard regression trees. However, having reached a leaf (that represents a partition) we use the respective training cases to obtain a kernel prediction for the query case. From the perspective of local modeling these *local regression trees* have two main advantages. Firstly, they provide a focusing effect, that avoids looking for the nearest training cases in all available training data. Instead we only use the cases within the respective partition, which has large computational efficiency advantages. Secondly, the regression tree can be seen as providing a rough, but comprehensible, description of the regression surface approximated by local regression trees.

4 Experimental Evaluation

This section describes a series of experiments designed with the goal of comparing local regression trees with kernel regression modeling. The goal of these experiments is to compare the predictive accuracy of kernel models and local regression trees, and also to assert the computational efficiency gains of the later. Regarding local regression trees we have used exactly the same local modeling settings as for kernel regression, the single difference being that one is applied in the leaves of the trees while the other uses the information of all training set. The experimental methodology used was a 10-fold cross validation (CV). The results that are shown are averages of 10 repetitions of 10-fold CV runs. The error of the models was measured by the mean squared error (MSE) between the predicted and truth values. Differences that can be considered statistically significant are marked by + signs (one sign means 95% confidence and two 99% confidence). The best results are presented in bold face.

Table 1 shows the results of these experiments with three different domains. *Close Nikkei 225* and *Close Dow Jones* consist of trying to predict the evolution of the Nikkei 225 and Dow Jones stock market indices for the next day based on information of previous days values and other indices. *Telecomm* is a commercial telecommunications problem used in a study by Weiss and Indurkhya [7]. The two former consist of 2399 observations each described by 50 input variables, while the later contains 15000 cases described by 48 variables.

Table 1. Comparing local regression trees with kernel models.

	Close Nikkei 225			*Close Dow Jones*			*Telecomm*		
	Local RT	*Kernel*		*Local RT*	*Kernel*		*Local RT*	*Kernel*	
MSE	140091.6	**125951.1**	++	**86.8**	214.5	++	**42.40**	57.19	++
CPU sec.	**4.4**	6.5	++	**2.47**	6.66	++	**63.57**	452.88	++

The results in terms of predictive accuracy are contradictory. In effect, both two methods achieve statistically significant (> 99% confidence) wins on different domains. However, local regression trees are able to significantly outperform kernel models in terms of computation efficiency, in spite of the small size of both the training and testing samples. In effect, additional simulation studies with increasing sample sizes have shown a more significant efficiency advantage of local regression trees [6]. Further details on these and other experiments can be found in [5, 6].

5 Conclusions

Local regression is a well-known data analysis method with excellent modeling abilities in a large range of problems. However, these techniques suffer from a high computational complexity and by not obtaining any visible and comprehensible model of the data. These can be considered major drawbacks in a typical data mining scenario.

In this paper we have described local regression trees that can be regarded as a new type of regression models that integrate a partition-based technique with local modeling. Local regression trees provide the smoothing effects of local modeling within the efficiency and comprehensibility of partition-based methods. Through the use of kernel models in the leaves of a standard regression tree we are able to provide a focusing effect on the use of kernel models with large advantages in the computation necessary to obtain the predictions. At the same time, the partitioning obtained with the tree can be regarded as a comprehensible overview of the regression surface being used to obtain the predictions.

We have carried out a large set of experiments that confirmed that local regression trees have an overwhelming advantage in terms of computation time with respect to standard local modeling techniques. Moreover, we have observed significant advantages in terms of predictive accuracy in several data sets.

References

1. Atkeson,C.G., Moore,A.W., Schaal,S.: Locally Weighted Learning. *Artificial Intelligence Review*, **11**, 11-73. Special issue on lazy learning, Aha, D. (Ed.), 1997.
2. Breiman,L. , Friedman,J.H., Olshen,R.A. & Stone,C.J.: *Classification and Regression Trees*. Wadsworth Int. Group, Belmont, California, USA, 1984.
3. Cleveland,W., Loader,C.: Smoothing by Local Regression: Principles and Methods (with discussion). *Computational Statistics*, 1995.
4. Fayyad,U.,Shapiro,G.,Smyth,P.:From data mining to knowledge discovery: an overview. In *Advances in Knowledge Discovery and Data Mining*, Fayyad *et al.*(eds). AAAI Press (1996).
5. Torgo, L.: *Inductive Learning of Tree-based Regression Models*. Ph.D. Thesis. Dept. of Computer Science, Faculty of Sciences. University of Porto, 1999. Available at http://www.ncc.up.pt/~ltorgo.
6. Torgo,L.: Efficient and Comprehensible Local Regression. LIACC, Machine Learning Group, Internal Report n.99.2 , 1999. Available at http://www.ncc.up.pt/~ltorgo.
7. Weiss, S. and Indurkhya, N.: Rule-based Machine Learning Methods for Functional Prediction. *Journal of Artificial Intelligence Research* (JAIR), **3**, pp.383-403, 1995

Information Granules for Spatial Reasoning

Andrzej Skowron[1], Jaroslaw Stepaniuk[2], and Shusaku Tsumoto[3]

[1] Institute of Mathematics, Warsaw University,
Banacha 2, 02-097 Warsaw, Poland,
skowron@mimuw.edu.pl
[2] Institute of Computer Science, Bialystok University of Technology,
Wiejska 45A, 15-351 Bialystok, Poland,
jstepan@ii.pb.bialystok.pl
[3] Department of Medical Informatics, Shimane Medical University
89-1 Enya-cho, Izumo-city, Shimane 693-8501 Japan
tsumoto@computer.org

Abstract. The aim of the paper is to present an outline of granular computing framework for spatial reasoning. In our previous papers we have discussed basic notions related to granular computing, namely the information granule syntax and semantics as well as the inclusion and closeness (similarity) relations of granules. Different information sources (units, agents) are equipped with two kinds of operations on information granules: operations possessed by agents transforming tuples of information granules into new granules and approximation operations for computing by agents information granule approximations delivered by other agents. More complex granules are constructed by means of these operations from some input information granules.

1 Motivation

We would like to discuss briefly an example showing a motivation for our work [10]. Let us consider a team of agents recognizing the situation on the road. The aim is to classify a given situation as, e.g., *dangerous or not.* This *soft specification granule* is represented by a family of information granules called *case soft patterns* representing cases, like *cars are too close.* The whole scene (actual situation on the road) is decomposed into regions perceived by local agents. Higher level agents can reason about regions observed by team of their children agents. They can express in their own languages features used by their children. Moreover, they can use new features like attributes describing relations between regions perceived by children agents. The problem is how to organize agents into a team having, e.g., tree structure, with the property that the information granules synthesized by the team from input granules (being local perceptions of local agents) will identify the situation on the road in the following sense: returned by the team granule is sufficiently close to the soft specification granule if and only if the situation on the road is dangerous and moreover, if any returned granule occurs to be sufficiently close to the specification granule then the relevant case soft pattern is identified. The aim of our project is to develop foundations

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 380–383, 2000.
© Springer-Verlag Berlin Heidelberg 2000

for this kind of reasoning. In particular it is necessary to give precise meaning to the notions like: information granules, soft information granules, closeness of information granules in satisfactory degree, information granules synthesized by team of agents etc. The presented paper and its extension [8] realize the first step towards this goal.

In Figure 1 the following entities are depicted:

- a specification soft granule represented by family of case soft granules g_1, g_2, g_3, g_4;
- input granules ig_1, ig_2 representing actual local situations for ag_1, ag_2;
- higher level granules describing situation received by fusion of granules perceived by ag_1, ag_2 taking into account the relationships between granules and a context in which they appear;
- og, og_1, og_2 granules returned by ag, ag_1, ag_2, respectively; og is received by performing an operation at ag on og_1, og_2.

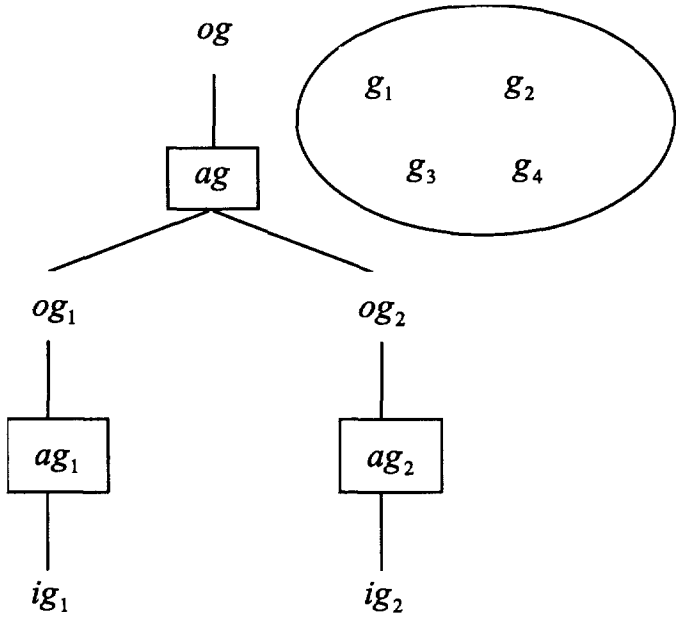


Fig. 1. Illustrative Example

To sum up, we consider a set of agents Ag. Each agent is equipped with some approximation spaces (defined using rough set approach [3]). Agents are cooperating to solve a problem specified by a special agent called *customer-agent.* The result of cooperation is a scheme of agents. In the simplest case the scheme

can be represented by a tree labeled by agents. In this tree leaves are delivering some information granules (representing of perception in a given situation by leaf agents) and any non-leaf agent $ag \in Ag$ is performing an operation $o(ag)$ on approximations of granules delivered by its children. The root agent returns an information granule being the result of computation by the scheme on granules delivered by leaf agents. It is important to note that different agents use different languages. Thus granules delivered by children agents to their father can be usually perceived by him in an approximate sense before he can perform any operation on delivered granules.

2 Information Granules

Methods for qualitative spatial reasoning [5], [2], [9], [1], [10] are closely related to a paradigm Computing with Words recently formulated by Lotfi Zadeh [11], [12]. Several attempts have been made to develop foundations for computing with words[12]. Among them there is a rapidly growing area of granular computing aiming to develop models for computing with information granules (see e.g. [4]).

They are two basic notions for granular computing: information granule and calculus on information granules [4].

Notions of information granule [11], [4] and information granule similarity (inclusion or closeness) are very useful for knowledge discovery. Informally speaking, information granules can be treated as linked collections of objects drawn together by the criteria of indiscernibility, similarity or functionality [11].

In [6], [7] several examples of complex information granules have been discussed. We have presented syntax, semantics, relations of inclusion ν_p and closeness cl_p for information granules and a general recursive scheme for construction of more complex granules from simpler ones. In particular, the inclusion and closeness relations for more complex granules are defined by extension of these relations for the granules being parts of those complex granules.

In this paper we elaborate a general scheme for information granule construction in distributed systems introduced in [7]. We describe only the main idea of our approach. The reader can find a more complete version in [8].

Teams of agents organized, e.g., along the schemes of decomposition of complex objects (representing situations on the road) into trees. The trees are represented by expressions called terms. Two granules are defined being values of t under the valuation val for any valuation val of leaf agents of a given term t in the set of input granules. They are called the lower and upper approximations of t under val. The necessity to consider rather approximation of granule returned by a given term t under a given valuation val than the exact value of t under val is a consequence of the mentioned above ability of agents to perceive in approximate sense only of information granules received from other agents. Similarity relations extracted from data allow to measure the closeness of these granules, in particular to the soft specification granule.

We consider problems of agent team (terms) synthesis for different tasks. For example, we are looking for a strategy returning for any valuation val (rep-

resenting global situation) a term (agent team) t with the following property: the lower and upper values of t under *val* are sufficiently close to a given soft specification granule if and only if the global situation represented by *val* really matches this specification.

We also emphasize [8] the problem of the robust granule construction. We use some ideas from rough mereology [4] to specify the rules describing the ranges in which parameters of granules being arguments of operations on granules can be changed to assure that the results of the operations on these granules are sufficiently close. We suggest that such rules should be extracted from data. The construction of such robust granules seems to be important for spatial reasoning.

Progress in solving the above discussed problems is strongly dependent on further results in foundations of granular computing including soft information granule understanding, methods for measuring of different kinds of information granule closeness or methods for information granule transformation.

Acknowledgments

This research was supported by the grants from the State Committee for Scientific Research (KBN) and the Research Grant of the European Union - ESPRIT–CRIT 2 No. 20288. Jaroslaw Stepaniuk has been supported by the KBN grant No. 8 T11C 023 15. Andrzej Skowron has been also partially supported by grant of the Wallenberg fundation.

References

1. Düntsch I., Wang H., McCloskey S.: Relations Algebras in Qualitative Spatial Reasoning, Fundamenta Informaticae **39(3)**, 1999, pp. 229–248.
2. Escrig M.T., Toledo F.: Qualitative Spatial Reasoning: Theory and Practice, IOS Press, Amsterdam, 1998.
3. Pawlak Z.: Rough Sets. Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Dordrecht, 1991.
4. Polkowski L., Skowron A.: Towards Adaptive Calculus of Granules, In: [12], vol.1, pp. 201–227.
5. Roddick J.F., Spiliopoulou M.: A Bibliography of Temporal, Spatial, and Temporal Data Mining Research, Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining **1(1)**, pp. 34–38.
6. Skowron A., Stepaniuk J.: Towards Discovery of Information Granules, Lecture Notes in Artificial Intelligence 1704, Springer-Verlag, 1999, pp. 542–547.
7. Skowron A., Stepaniuk J.: Information Granules in Distributed Environment, Lecture Notes in Artificial Intelligence 1711, Springer-Verlag, 1999, pp. 357–365.
8. Skowron A., Stepaniuk J., Tsumoto S.: Information Granules for Spatial Reasoning, Bulletin of International Rough Set Society **3/4** 1999, pp. 147–154.
9. Sogo T., Ishiguro H., Ishida T.: Acquisition of Qualitative Spatial Representation by Visual Observation, IJCAI 1999, pp. 1054–1060.
10. WWW SPACENET page: http://agora.scs.leeds.ac.uk/spacenet/.
11. Zadeh L.A.: Fuzzy Logic = Computing with Words, IEEE Trans. on Fuzzy Systems Vol. 4, 1996, pp. 103–111.
12. Zadeh L.A., Kacprzyk J. (Eds.): Computing with Words in Information/Intelligent Systems vol.1–2, Physica–Verlag, Heidelberg, 1999.

Uncovering the Hierarchical Structure of Text Archives by Using an Unsupervised Neural Network with Adaptive Architecture

Dieter Merkl and Andreas Rauber

Institut für Softwaretechnik, Technische Universität Wien
Favoritenstraße 9–11/188, A–1040 Wien, Austria
www.ifs.tuwien.ac.at/~dieter www.ifs.tuwien.ac.at/~andi

Abstract. Discovering the inherent structure in data has become one of the major challenges in data mining applications. It requires the development of stable and adaptive models that are capable of handling the typically very high-dimensional feature spaces. In this paper we present the *Growing Hierarchical Self-Organizing Map (GH-SOM)*, a neural network model based on the self-organizing map. The main feature of this extended model is its capability of growing both in terms of map size as well as in a three-dimensional tree-structure in order to represent the hierarchical structure present in a data collection. This capability, combined with the stability of the self-organizing map for high-dimensional feature space representation, makes it an ideal tool for data analysis and exploration. We demonstrate the potential of this method with an application from the information retrieval domain, which is prototypical of the high-dimensional feature spaces frequently encountered in today's applications.

1 Introduction

Today's information age may be characterized by constant massive production and dissemination of written information. More powerful tools for exploring, searching, and organizing the available mass of information are needed to cope with this situation. An attractive way to assist the user in document archive exploration is based on unsupervised artificial neural networks, especially self-organizing maps [3], for document space representation. A number of research publications show that this idea has found appreciation in the community [4, 5, 6, 7, 9, 13]. Self-organizing maps are used to visualize the similarity between documents in terms of distances within the two-dimensional map display. Hence, similar documents may be found in neighboring regions of the map.

Despite the large number of research reports on self-organizing map usage for document archive representation, some difficulties remain untouched. First, the determination of a suitable number of neurons requires some insight into the structure of the document archive. This cannot be assumed, however, in case of unknown document collections. Thus, it might be helpful if the neural network would be able to determine this number during its learning process. Second, hierarchical relations between the input data are not mirrored in a straight-forward

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 384-395, 2000.
© Springer-Verlag Berlin Heidelberg 2000

manner. Obviously, we should expect such hierarchical relations in document collections where different subject matters are covered. The identification of these hierarchical relations remains a highly important data mining task that cannot be addressed conveniently within the framework of self-organizing map usage.

In order to overcome these two limitations of self-organizing maps we propose a novel neural network architecture in this paper, i.e. the *growing hierarchical self-organizing map*, *GH-SOM* for short. This neural network architecture is capable of determining the required number of units during its unsupervised learning process. Additionally, the data set is clustered hierarchically by relying on a layered architecture comprising a number of independent self-organizing maps within each layer.

The remainder of this paper is organized as follows. In Section 2 we provide an outline of architecture and learning rule of the *growing hierarchical self-organizing map*. Section 3 gives a description of the experimental data set, namely a collection of articles from the *Time Magazine*. We provide results from using both the self-organizing map and the *growing hierarchical self-organizing map* with this data set in Section 4. Related work is briefly described in Section 5. Finally, we present our conclusions in Section 6.

2 Growing Hierarchical Self-Organizing Maps

The key idea of the *growing hierarchical self-organizing map* (*GH-SOM*) is to use a hierarchical neural network structure composed of a number of individual layers each of which consists of independent *self-organizing maps* (*SOMs*). In particular, the neural network architecture starts with a single-unit *SOM* at layer 0. One *SOM* is used at layer 1 of the hierarchy. For every unit in this layer 1 map, a *SOM* might be added to the next layer of the hierarchy. This principle is repeated with the third and any further layers of the *GH-SOM*.

Since one of the shortcomings of *SOM* usage is its fixed network architecture in terms of the number of units and their arrangement, we rather rely on an incrementally growing version of the *SOM*. This relieves us from the burden of predefining the network's size which is now determined during the unsupervised training process according to the peculiarities of the input data space. Pragmatically speaking, the *GH-SOM* is intended to uncover the hierarchical relationship between input data in a straight-forward fashion. More precisely, the similarities of the input data are shown in increasingly finer levels of detail along the hierarchy defined by the neural network architecture. *SOMs* at higher layers give a coarse grained picture of the input data space whereas *SOMs* of deeper layers provide fine grained input discrimination. The growth process of the neural network is guided by the so-called *quantization error* which is a measure of the quality of input data representation.

The starting point for the growth process is the overall deviation of the input data as measured with the single-unit *SOM* at layer 0. This unit is assigned a weight vector m_0, $m_0 = [\mu_{0_1}, \mu_{0_2}, \ldots, \mu_{0_n}]^T$, computed as the average of all input data. The deviation of the input data, i.e. the *mean quantization error* of

this single unit, is computed as given in Expression (1) with d representing the number of input data x. We will refer to the *mean quantization error* of a unit as **mqe** in lower case letters.

$$\mathbf{mqe}_0 = \frac{1}{d} \cdot ||m_0 - x|| \tag{1}$$

After the computation of $\mathbf{mqe}_0$, training of the *GH-SOM* starts with its first layer *SOM*. This first layer map initially consists of a rather small number of units, e.g. a grid of 2×2 units. Each of these units i is assigned an n-dimensional weight vector m_i, $m_i = [\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_n}]^T$, $m_i \in \Re^n$, which is initialized with random values. It is important to note that the weight vectors have the same dimensionality as the input patterns.

The learning process of *SOMs* may be described as a competition among the units to represent the input patterns. The unit with the weight vector being closest to the presented input pattern in terms of the input space wins the competition. The weight vector of the winner as well as units in the vicinity of the winner are adapted in such a way as to resemble more closely the input pattern.

The degree of adaptation is guided by means of a learning-rate parameter α, decreasing in time. The number of units that are subject to adaptation also decreases in time such that at the beginning of the learning process a large number of units around the winner is adapted, whereas towards the end only the winner is adapted. These units are chosen by means of a neighborhood function h_{ci} which is based on the units' distances to the winner as measured in the two-dimensional grid formed by the neural network. In combining these principles of *SOM* training, we may write the learning rule as given in Expression (2), where x represents the current input pattern, and c refers to the winner at iteration t.

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot h_{ci}(t) \cdot [x(t) - m_i(t)] \tag{2}$$

In order to adapt the size of this first layer *SOM*, the *mean quantization error* of the map is computed ever after a fixed number λ of training iterations as given in Expression (3). In this formula, u refers to the number of units i contained in the *SOM* m. In analogy to Expression (1), $\mathbf{mqe}_i$ is computed as the average distance between weight vector m_i and the input patterns mapped onto unit i. We will refer to the *mean quantization error* of a map as **MQE** in upper case letters.

$$\mathbf{MQE}_m = \frac{1}{u} \cdot \sum_i \mathbf{mqe}_i \tag{3}$$

The basic idea is that each layer of the *GH-SOM* is responsible for explaining some portion of the deviation of the input data as present in its preceding layer. This is done by adding units to the *SOMs* on each layer until a suitable size of the map is reached. More precisely, the *SOMs* on each layer are allowed to grow until the deviation present in the unit of its preceding layer is reduced to at least

a fixed percentage τ_m. Obviously, the smaller the parameter τ_m is chosen the larger will be the size of the emerging *SOM*. Thus, as long as $\mathbf{MQE}_m \geq \tau_m \cdot \mathbf{mqe}_0$ holds true for the first layer map m, either a new row or a new column of units is added to this *SOM*. This insertion is performed neighboring the unit e with the highest *mean quantization error*, $\mathbf{mqe}_e$, after λ training iterations. We will refer to this unit as the *error unit*. The distinction whether a new row or a new column is inserted is guided by the location of the most dissimilar neighboring unit to the *error unit*. Similarity is measured in the input space. Hence, we insert a new row or a new column depending on the position of the neighbor with the most dissimilar weight vector. The initialization of the weight vectors of the new units is simply performed as the average of the weight vectors of the existing neighbors. After the insertion, the learning-rate parameter α and the neighborhood function h_{ci} are reset to their initial values and training continues according to the standard training process of *SOMs*. Note that we currently use the same value of the parameter τ_m for each map in each layer of the *GH-SOM*. It might be subject to further research, however, to search for alternative strategies, where layer or even map-dependent quantization error reduction parameters are utilized.

Consider Figure 1 for a graphical representation of the insertion of units. In this figure the architecture of the *SOM* prior to insertion is shown on the left-hand side where we find a map of 2×3 units with the *error unit* labeled by e and its most dissimilar neighbor signified by d. Since the most dissimilar neighbor belongs to another row within the grid, a new row is inserted between units e and d. The resulting architecture is shown on the right-hand side of the figure as a map of now 3×3 units.

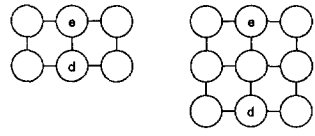


Fig. 1. Insertion of units to a self-organizing map

As soon as the growth process of the first layer map is finished, i.e. $\mathbf{MQE}_m < \tau_m \cdot \mathbf{mqe}_0$, the units of this map are examined for expansion on the second layer. In particular, those units that have a large *mean quantization error* will add a new *SOM* to the second layer of the *GH-SOM*. The selection of these units is based on the *mean quantization error* of layer 0. A parameter τ_u is used to describe the desired level of granularity in input data discrimination in the final maps. More precisely, each unit i fulfilling the criterion given in Expression (4) will be subject to hierarchical expansion.

$$\mathbf{mqe}_i > \tau_u \cdot \mathbf{mqe}_0 \tag{4}$$

The training process and unit insertion procedure now continues with these newly established *SOMs*. The major difference to the training process of the second layer map is that now only that fraction of the input data is selected for training which is represented by the corresponding first layer unit. The strategy for row or column insertion as well as the termination criterion is essentially the same as used for the first layer map. The same procedure is applied for any subsequent layers of the *GH-SOM*.

The training process of the *GH-SOM* is terminated when no more units require further expansion. Note that this training process does not necessarily lead to a balanced hierarchy, i.e. a hierarchy with equal depth in each branch. Rather, the specific requirements of the input data is mirrored in that clusters might exist that are more structured than others and thus need deeper branching. Consider Figure 2 for a graphical representation of a trained *GH-SOM*. In particular, the neural network depicted in this figure consists of a single-unit *SOM* at layer 0, a *SOM* of 2×3 units in layer 1, six *SOMs* in layer 2, i.e. one for each unit in the layer 1 map. Note that each of these maps might have a different number and different arrangements of units as shown in the figure. Finally, we have one *SOM* in layer 3 which was expanded from one of the layer 2 units.

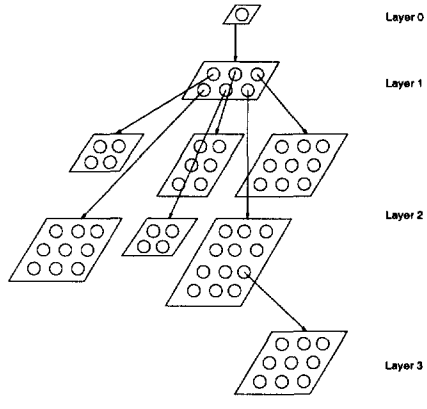


Fig. 2. Architecture of a trained *GH-SOM*

To summarize, the growth process of the *GH-SOM* is guided by two parameters τ_u and τ_m. The parameter τ_u specifies the desired quality of input data representation at the end of the training process. Each unit i with $\mathbf{mqe}_i > \tau_u \cdot \mathbf{mqe}_0$ will be expanded, i.e. a map is added to the next layer of the hierarchy, in order to explain the input data in more detail. Contrary to that, the parameter τ_m specifies the desired level of detail that is to be shown in a particular *SOM*. In other words, new units are added to a *SOM* until the **MQE** of the map is a certain fraction, τ_m, of the **mqe** of its preceding unit. Hence, the smaller τ_m the larger will be the emerging maps. Conversely, the larger τ_m the deeper will be the hierarchy.

3 Data Set

In the experiments presented hereafter we use the *TIME Magazine* article collection available at http://www.ifs.tuwien.ac.at/ifs/research/ir as a reference document archive. The collection comprises 420 documents from the *TIME Magazine* of the early 1960's. The documents can be thought of as forming topical clusters in the high-dimensional feature space spanned by the words that the documents are made up of. The goal is to map and identify those clusters on the 2-dimensional map display. Thus, we use full-text indexing to represent the various documents according to the vector space model of information retrieval. The indexing process identified 5923 content terms, i.e. terms used for document representation, by omitting words that appear in more than 90% or less than 1% of the documents. The terms are roughly stemmed and weighted according to a $tf \times idf$, i.e. term frequency $\times$ inverse document frequency, weighting scheme [14], which assigns high values to terms that are considered important in describing the contents of a document. Following the feature extraction process we end up with 420 vectors describing the documents in the 5923-dimensional document space, which are further used for neural network training.

4 Experimental Results

Figure 3 shows a conventional self-organizing map trained with the Times Article Collection data set. It consists of 10×15 units represented as table cells with a number of articles being mapped onto each individual unit. The articles mapped onto the same or neighboring units are considered to be similar to each other in terms of the topic they deal with. Due to space considerations we cannot present all the articles in the collection. We thus selected a number of units for detailed discussion.

We find, that the *SOM* has succeeded in creating a topology preserving representation of the topical clusters of articles. For example, in the lower left corner we find a group of units representing articles on the conflict in Vietnam. To name just a few, we find articles *T320, T369* on unit (14/1)[1], *T390, T418, T434* on unit (15/1) or *T390, T418, T434* on unit (15/2) dealing with the government crackdown on buddhist monks, next to a number of articles on units (15/4), (15/5) and neighboring ones, covering the fighting and suffering during the Vietnam War.

A cluster of documents covering affairs in the Middle-East is located in the lower right corner of the map around unit (15/10), next to a cluster on the so-called Profumo-Keeler affair, a political scandal in Great Britain in the 1960's, on and around units (11/10) and (12/10). Above this area, on units (6/10) and neighboring ones we find articles on elections in Italy and possible coalitions, next to two units (3/10) and (4/10) covering elections in India. Similarly, all other units on the map can be identified to represent a topical cluster of news

[1] We use the notion (x/y) to refer to the unit located in row x and column y of the map, starting with (1/1) in the upper left corner

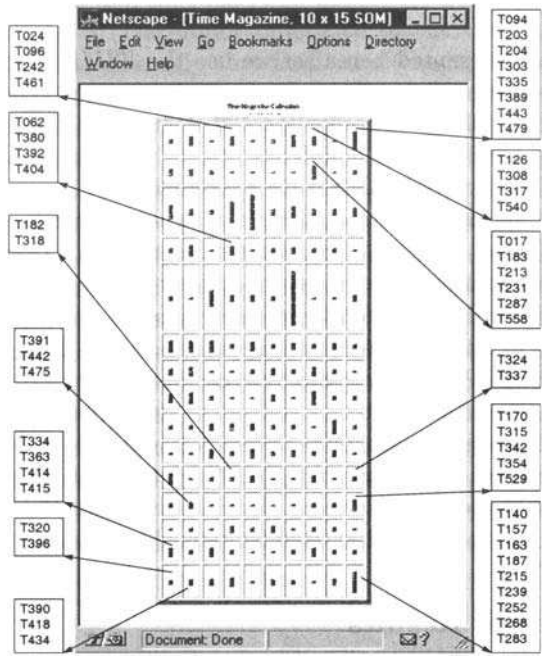


Fig. 3. 10 × 15 *SOM* of the *Time Magazine* collection

articles. For a more detailed discussion of the articles and topic clusters found on this map, we refer to [12] and the online-version of this map available at http://www.ifs.tuwien.ac.at/ifs/research/ir.

While we find the *SOM* to provide a good topologically ordered representation of the various topics found in the article collection, no information about topical hierarchies can be identified from the resulting flat map. Apart from this we find the size of the map to be quite large with respect to the number of topics identified. This is mainly due to the fact that the size of the map has to be determined in advance, before any information about the number of topical clusters is available.

To overcome these shortcomings we trained a growing hierarchical SOM. Based on the artificial unit representing the means of all data points at layer 0, the GH-SOM training algorithm started with a 2 × 2 SOM at layer 1. The training process for this map continued with additional units being added until the quantization error fell below a certain percentage of the overall quantization error of the unit at layer 0. The resulting first-layer map is depicted in Figure 4. The map has grown for two stages, adding one row and one column respectively, resulting in 3 × 3 units representing 9 major topics in the document collection.

For convenience we list the topics of the various units, rather then the individual articles in the figure. For example, we find unit (1/1) to represent all articles related to the situation in Vietnam, whereas Middle-East topics are cov-

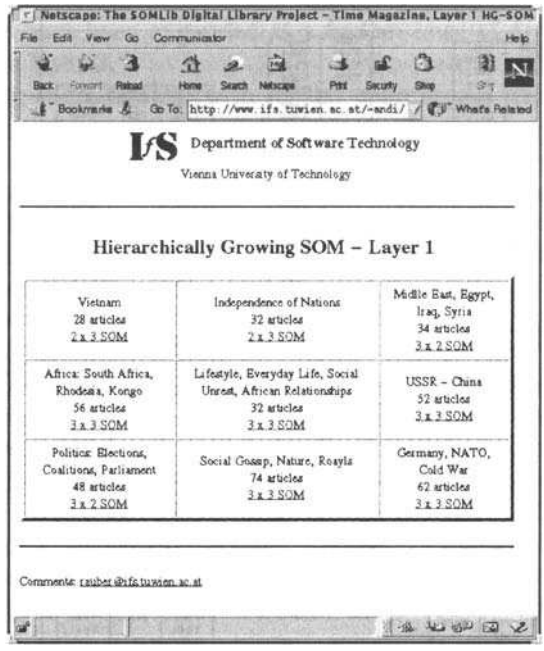


Fig. 4. Layer 1 of the *GH-SOM*

ered on unit (1/3), or articles related to elections and other political topics on unit (3/1) in the lower left corner to name but a few.

Based on this first separation of the most dominant topical clusters in the article collection, further maps were automatically trained to represent the various topics in more detail. This results in 9 individual maps on layer 2, each representing the data of the respective higher-layer unit in more detail. Some of the units on these layer 2 maps were further expanded as distinct *SOMs* in layer 3.

The resulting layer 2 maps are depicted in Figure 5. Please note, that – according to the structure of the data – the maps on the second layer have grown to different sizes, such as a small 2×2 map representing the articles of unit (3/1) of the first map, up to 3×3 maps for the units (2/1), (3/2) and (3/3). Taking a more detailed look at the first map of layer 2 representing unit (1/1) of layer 1 we find it to give a clearer representation of articles covering the situation in Vietnam. Units (1/1) and (2/1) on this map represent articles on the fighting during the Vietnam War, whereas the remaining units represent articles on the internal conflict between the catholic government and buddhist monks. At this layer, the two units (1/2) and (3/2) have further been expanded to form separate maps with 3×3 units each at layer 3. These again represent articles on the war and the internal situation in Vietnam in more detail.

To give another example of the hierarchical structures identified during the *growing hierarchical SOM* training process, we may take a look at the 2×3

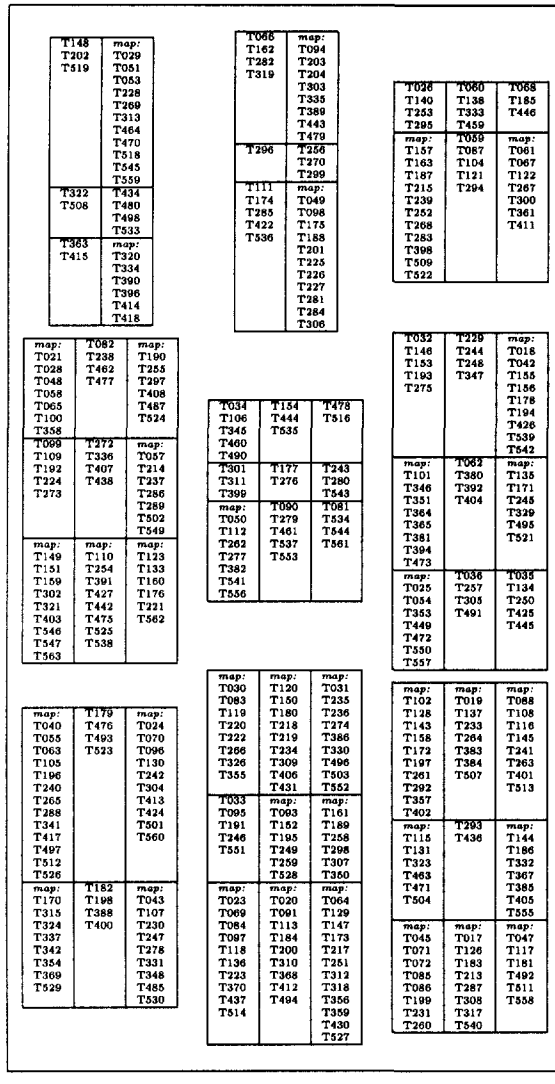


Fig. 5. Layer 2 of the *GH-SOM*: 1 *SOM* per unit of layer 1 *SOM*

map representing the articles of unit (3/1) of the first layer map. All of these articles were found to deal with political matters on layer 1. This common topic is now displayed in more detail at the resulting second-layer map. For example, we find unit (1/3) to represent articles on the elections in India. Next to these, we find on units (1/2) and (2/3) articles covering the elections and discussions about political coalitions between socialists and christian democrats in Italy. The remaining 3 units on this map deal with different issues related to the Profumo-Keeler scandal in Great Britain, covering the political hearings in parliament as

well as background information on this scandal and the persons involved. Again, some of the units have been expanded at a further level of detail forming 3×2 or 3×3 SOMs on layer 3.

For comparing the *GH-SOM* with its flat counterpart we may identify the locations of the articles on the 9 second-layer maps on the corresponding 10×15 *SOM*. This allows us to view the hierarchical structure of the data on the flat map. We find that, for example, the cluster on Vietnam simply forms one larger coherent cluster on the flat map in the lower left corner of the map covering the rectangle spanned by the units (14/1) and (15/5). The same applies to the cluster of Middle-East affairs, which is represented by the map of unit (1/3) in the *growing hierarchical SOM*. This cluster is mainly located in the lower right corner of the flat *SOM*. The cluster of political affairs, represented by unit (3/1) on the first layer of the *GH-SOM* and represented in more detail on its subsequent layers, is spread across the right side of the flat *SOM*, covering more or less all units on columns 9 and 10 and between rows 3 and 12. Note, that this common topic of political issues is not easily discernible from the overall map representation in the flat *SOM*, where exactly this hierarchical information is lost. The subdivision of this cluster on political matters becomes further evident when we consider the second layer classification of this topic area, where the various sub-topics are clearly separated, covering Indian elections, Italian coalitions and the British Profumo-Keeler scandal.

As another interesting feature of the *GH-SOM* we want to emphasize on is the overall reduction in map size. During analysis we found the second layer of the *GH-SOM* to represent the data at about the same level of topical detail as the corresponding flat *SOM*. Yet the number of units of all individual second-layer *SOMs* combined is only 87 as opposed to 150 units in the flat 10×15 *SOM*. Of course we might decide to train a smaller flat *SOM* of, say 9×10 units. However, with the *GH-SOM* model, this number of units is determined automatically, and only the necessary number of units is created for each level of detail representation required by the respective layer. Furthermore, not all branches are grown to the same depth of the hierarchy. As can be seen from Figure 5, only some of the units are further expanded in a layer 3 map. With the resulting maps at all layers of the hierarchy being rather small, activation calculation and winner evaluation is by orders of magnitude faster than in the conventional model. Apart from the speedup gained by the reduced network size, orientation for the user is highly improved as compared to the rather huge maps which can not be easily comprehended as a whole.

5 Related Work

A number of extensions and modifications have been proposed over the years in order to enhance the applicability of *SOMs* to data mining, specifically cluster identification. Some of the approaches, such as the *U-Matrix* [15], or the *Adaptive Coordinates* and *Cluster Connection* techniques [8] focus on the detection and visualization of clusters in conventional *SOMs*. Similar cluster information can

also be obtained using our *LabelSOM* method [11], which automatically describes the characteristics of the various units. Grouping units that have the same descriptive keywords assigned to them allows to identify topical clusters within the *SOM* map area. However, none of the methods identified above facilitates the detection of hierarchical structure inherent in the data.

The *hierarchical feature map* [10] addresses this problems by modifying the *SOM* network architecture. Instead of training a flat *SOM* map, a balanced hierarchical structure of *SOMs* is trained. Similar to our *GH-SOM* model, the data mapped onto one single unit is represented at some further level of detail in the lower-level map assigned to this unit. However, this model rather pretends to represent the data in a hierarchical way rather than really reflecting the structure of the data. This is due to the fact that the architecture of the network has to be defined in advance, i.e. the number of layers and the size of the maps at each layer is fixed prior to network training. This leads to the definition of a balanced tree which is used to represent the data. What we want, however, is a network architecture definition based on the actual data presented to the network. This requires the *SOM* to actually use the data available to define its architecture, the required levels in the hierarchy and the size of the map at each level, none of which is present in the *hierarchical feature map* model.

The necessity of having to define the size of the *SOM* in advance has been addressed in several models, such as the *Incremental Grid Growing* [1] or *Growing Grid* [2] models. The latter, similar to our *GH-SOM* model, adds rows and columns during the training process, starting with an initial 2×2 SOM. However, the main focus of this model lies with an equal distribution of input signals across the map, adding units in the neighborhood of units that represent an unproportionally high number of data points. It does thus not primarily reflect the concept of representation at a certain level of detail, which is rather expressed in the overall quantization error rather then the number of data points mapped onto certain areas. The *Incremental Grid Growing* model, on the other hand, can add new units only on the borders of the map. Neither of this models, however, takes the inherently hierarchical structure of data into account.

6 Conclusions

We have presented the *Growing Hierarchical Self-Organizing Map* (*GH-SOM*), a neural network based on the self-organizing map (*SOM*), a model that has proven to be effective for cluster analysis of very high-dimensional feature spaces. Its main benefits are due to the model's capabilities to (1) determine the number of neural processing units required in order to represent the data at a desired level of detail and (b) to create a network architecture reflecting the hierarchical structure of the data. The resulting benefits are numerous: first, the processing time is largely reduced by training only the necessary number of units for a certain degree of detail representation. Second, the *GH-SOM* by its very architecture resembles the hierarchical structure of data, allowing the user to understand and analyze large amounts of data in an explorative way. Third, with the vari-

ous emergent maps at each level in the hierarchy being rather small, it is easier for the user to keep an overview of the various clusters identified in the data and to build a cognitive model of it in a very high-dimensional feature space. We have demonstrated the capabilities of this approach by an application from the information retrieval domain, where text documents, which are located in a high-dimensional feature space spanned by the words in the documents, are clustered by their mutual similarity and where the hierarchical structure of these documents is reflected in the resulting network architecture.

References

1. J. Blackmore and R. Miikkulainen. Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map. In *Proc. of the IEEE Int'l. Conf. on Neural Networks (ICNN'93)*, San Francisco, CA, USA, 1993.
2. B. Fritzke. Growing grid – a self-organizing network with constant neighborhood range and adaption strength. *Neural Processing Letters*, 2, No. 5:1 – 5, 1995.
3. T. Kohonen. *Self-Organizing Maps.* Springer Verlag, Berlin, Germany, 1995.
4. T. Kohonen. Self-organization of very large document collections: State of the art. In *Proc Int'l Conf on Artificial Neural Networks*, Skövde, Sweden, 1998.
5. X. Lin, D. Soergel, and G. Marchionini. A self-organizing semantic map for information retrieval. In *Proc. Int'l ACM SIGIR Conf. on R & D in Information Retrieval*, Chicago, IL, 1991.
6. D. Merkl. Text classification with self-organizing maps: Some lessons learned. *Neurocomputing*, 21(1–3), 1998.
7. D. Merkl. Text data mining. In *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text.* Marcel Dekker, New York, 1998.
8. D. Merkl and A. Rauber. Alternative ways for cluster visualization in self-organizing maps. In *Proc. of the Workshop on Self-Organizing Maps (WSOM97)*, Helsinki, Finland, 1997.
9. D. Merkl and A. Rauber. Uncovering associations between documents. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI99)*, Stockholm, Sweden, 1999.
10. R. Miikkulainen. Script recognition with hierarchical feature maps. *Connection Science*, 2:83 – 101, 1990.
11. A. Rauber and D. Merkl. Automatic labeling of self-organizing maps: Making a treasure map reveal its secrets. In *Proc. 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD99)*, Beijing, China, 1999. Springer Verlag.
12. A. Rauber and D. Merkl. Using self-organizing maps to organize document collections and to characterize subject matters: How to make a map tell the news of the world. In *Proc. 10th Int'l Conf. on Database and Expert Systems Applications (DEXA99)*, Florence, Italy, 1999.
13. D. Roussinov and M. Ramsey. Information forage through adaptive visualization. In *Proc. ACM Conf. on Digital Libraries 98 (DL98)*, Pittsburgh, PA, USA, 1998.
14. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1989.
15. A. Ultsch. Self-organizing neural networks for visualization and classification. In *Information and Classification. Concepts, Methods and Application.* Springer Verlag, 1993.

Mining Access Patterns Efficiently from Web Logs *

Jian Pei, Jiawei Han, Behzad Mortazavi-asl, and Hua Zhu

School of Computing Science, Simon Fraser University, Canada
{peijian,han,mortazav,hzhua}@cs.sfu.ca

Abstract. With the explosive growth of data available on the World Wide Web, discovery and analysis of useful information from the World Wide Web becomes a practical necessity. Web access pattern, which is the sequence of accesses pursued by users frequently, is a kind of interesting and useful knowledge in practice.
In this paper, we study the problem of mining access patterns from Web logs efficiently. A novel data structure, called **Web access pattern tree**, or WAP-tree in short, is developed for efficient mining of access patterns from pieces of logs. The Web access pattern tree stores highly compressed, critical information for access pattern mining and facilitates the development of novel algorithms for mining access patterns in large set of log pieces. Our algorithm can find access patterns from Web logs quite efficiently. The experimental and performance studies show that our method is in general an order of magnitude faster than conventional methods.

1 Introduction

With the explosive growth of data available on the World Wide Web, discovery and analysis of useful information from the World Wide Web becomes a practical necessity. *Web mining* is the application of data mining technologies to huge Web data repositories. Basically, there are two domains that pertain to Web mining: Web content mining and Web usage mining. The former is the process of extracting knowledge from the content of Web sites, whereas the latter, also known as *Web log mining*, is the process of extracting interesting patterns in Web access logs.

Web servers register a Web log entry for every single access they get, in which important pieces of information about accessing are recorded, including the URL requested, the IP address from which the request originated, and a timestamp. A fragment of log file is shown as follows.

```
pm21s15.intergate.bc.ca - - [06/Oct/1999:00:00:09 -0700] "GET / HTTP/1.1" 200 5258
"http://www.sfu.ca/academic_programs.htm" "Mozilla/4.0 (compatible; MSIE 4.01; Windows 95)"

pm21s15.intergate.bc.ca - - [06/Oct/1999:00:00:11 -0700] "GET /images/bullets/bsqs.gif
HTTP/1.1" 200 489 "http://www.cs.sfu.ca/" "Mozilla/4.0 (compatible; MSIE 4.01; Windows 95)"
```

* The work was supported in part by the Natural Sciences and Engineering Research Council of Canada (grant NSERC-A3723), the Networks of Centres of Excellence of Canada (grant NCE/IRIS-3), and the Hewlett-Packard Lab.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 396–407, 2000.
© Springer-Verlag Berlin Heidelberg 2000

There are many efforts towards mining various patterns from Web logs, e.g. [4,11,15]. Web access patterns mined from Web logs are interesting and useful knowledge in practice. Examples of applications of such knowledge include improving designs of web sites, analyzing system performance as well as network communications, understanding user reaction and motivation, and building adaptive Web sites [5,10,13,14].

Essentially, a Web access pattern is a sequential pattern in a large set of pieces of Web logs, which is pursued frequently by users. Some research efforts try to employ techniques of sequential pattern mining [2], which is mostly based on association rule mining [1], for discovering Web access patterns from Web logs.

Sequential pattern mining, which discovers frequent patterns in a sequence database, was first introduced by Agrawal and Srikant [2] as follows: *given a sequence database where each sequence is a list of transactions ordered by transaction time and each transaction consists of a set of items, find all sequential patterns with a user-specified minimum support, where the support is the number of data sequences that contain the pattern.*

Since its introduction, there have been many studies on efficient mining techniques and extensions of sequential pattern mining method to mining other time-related frequent patterns [2,12,8,7,3,9,6].

Srikant and Agrawal [12] generalized their definition of sequential patterns in [2] to include time constraints, sliding time window, and user-defined taxonomy and developed a generalized sequential pattern mining algorithm, GSP, which outperforms their AprioriAll algorithm [2]. GSP mines sequential patterns by scanning the sequence database multiple times. In the first scan, it finds all frequent 1-items and forms a set of 1-element frequent sequences. In the following scans, it generates (step-wise longer) *candidate sequences* from the set of frequent sequences and check their supports. GSP is efficient when the sequences are not long as well as the transactions are not large. However, when the length of sequences increase and/or when the transactions are large, the number of candidate sequences generated may grow exponentially, and GSP will encounter difficulties.

All of the above studies on time-related (sequential or periodic) frequent pattern mining adopt an Apriori like paradigm, which promotes a generate-and-test method: first generate a set of candidate patterns and then test whether each candidate may have sufficient support in the database (i.e., passing the minimum support threshold test). The Apriori heuristic is on how to generate a *reduced set* of candidates at each iteration.

However, as these algorithms are level-wise, Apriori -like in nature, they encounter the same difficulty when the length of the pattern grows long, which is exactly the case in Web access pattern mining. In Web log mining, the length of Web log pieces can be pretty long, while the number of such pieces can be quite huge in practice.

In this paper, we investigate the issues related to efficiently mining Web access from large set of pieces of Web log. The main contributions are as follows. First,

a concise, highly compressed WAP-tree structure is designed and implemented which handles the sequences elegantly. Second, an efficient mining algorithm, WAP-mine , is developed for mining the complete (but nonredundant) Web access patterns from large set of pieces of Web log. Third, a performance study has been conducted which demonstrates that the WAP-mine algorithm is an order of magnitude faster than its Apriori -based counterpart for mining Web access patterns.

The remaining of the paper is organized as follows. The problem is defined in Section 2, while the general idea of our novel method is presented in Section 3. Section 4 and 5 focus on construction WAP-tree and mining the tree, respectively. We show the experimental results and conclude the paper in Section 6.

2 Problem Statement

In this paper, we focus on *mining Web access patterns*. In general, a Web log can be regarded as a sequence of pairs of user identifier and event. In this investigation, Web log files are divided into pieces per mining purpose. Preprocessing can be applied to the original Web log files, so that pieces of Web logs can be obtained. Each piece of Web log is a sequence of events from one user or session in timestamp ascending order, i.e. event happened early goes first. We model pieces of Web logs as sequences of events, and mine the sequential patterns over certain support threshold.

Let E be a set of **events**. A **Web log piece** or **(Web) access sequence** $S = e_1e_2\cdots e_n$ $(e_i \in E)$ for $(1 \leq i \leq n)$ is a sequence of events, while n is called the **length** of the access sequence. An access sequence with length n is also called an n-**sequence**. Please note that it is not necessary that $e_i \neq e_j$ for $(i \neq j)$ in an access sequence S. Repetition is allowed. For example, aab and ab are two different access sequences, in which a and b are two events.

Access sequence $S' = e'_1e'_2\cdots e'_l$ is called a **subsequence** of access sequence $S = e_1e_2\cdots e_n$, and S a **super-sequence** of S', denoted as $S' \subseteq S$, if and only if there exist $1 \leq i_1 < i_2 < \cdots < i_l \leq n$, such that $e'_j = e_{i_j}$ for $(1 \leq j \leq l)$. Access sequence S' is a **proper subsequence** of sequence S, denoted as $S' \subset S$, if and only if S' is a subsequence of S and $S' \neq S$.

In access sequence $S = e_1e_2\cdots e_ke_{k+1}\cdots e_n$, if subsequence $S_{suffix} = e_{k+1}\cdots e_n$ is a super sequence of pattern $P = e'_1e'_2\cdots e'_l$, and $e_{k+1} = e'_1$, the subsequence of S, $S_{prefix} = e_1e_2\cdots e_k$, is called the **prefix** of S with respect to pattern P.

Given a set of access sequence $\mathcal{WAS} = \{S_1, S_2, \ldots, S_m\}$, called **Web access sequence database**, in which S_i $(1 \leq i \leq m)$ are access sequences. The **support** of access sequence S in $\mathcal{WAS}$ is defined as $sup_{\mathcal{WAS}}(S) = \frac{|\{S_i | S \subseteq S_i\}|}{m}$. $sup_{\mathcal{WAS}}(S)$ is also denoted as $sup(S)$ if $\mathcal{WAS}$ is clear from the context. A sequence S is said a ξ-**pattern** or simply **(Web) access pattern** of $\mathcal{WAS}$, if $sup_{\mathcal{WAS}}(S) \geq \xi$. Please note that the access sequences in a Web access sequence database need not be of the same length. Although events can be repeated in an access sequence or pattern, any pattern can get support at most once from one access sequence.

Problem Statement. The problem of **Web access pattern mining** is: *given Web access sequence database $\mathcal{WAS}$ and a support threshold ξ, mine the complete set of ξ-patterns of $\mathcal{WAS}$.*

Example 1. Let $\{a, b, c, d, e, f\}$ be a set of events, and 100, 200, 300, and 400 are identifiers of users. A fragment of Web log records the information as follows.

$$\langle 100, a\rangle\langle 100, b\rangle\langle 200, a\rangle\langle 300, b\rangle\langle 200, b\rangle\langle 400, a\rangle\langle 100, a\rangle\langle 400, b\rangle\langle 300, a\rangle\langle 100, c\rangle$$

$$\langle 200, c\rangle\langle 400, a\rangle\langle 200, a\rangle\langle 300, b\rangle\langle 200, c\rangle\langle 400, c\rangle\langle 400, c\rangle\langle 300, a\rangle\langle 300, c\rangle$$

A preprocessing which divides the log file into access sequences of individual users is applied to the log file, while the resulting access sequence database, denoted as $\mathcal{WAS}$, is shown in the first two columns in Table 1.

There are totally 4 access sequences in the database. They are not with same length. The first access sequence, *abdac*, is a 5-sequence, while *ab* is a subsequence of it. In access sequence of user 200, both *e* and *eaebc* are prefixes with respect to *ac*. fc is a 50%-pattern because it gets supports from access sequence of user 300 and 400. Please note that even fc appears twice in the access sequence of user 400, $afbacfc$, but the sequence contributes only one to the count of fc.

Table 1. A database of Web access sequences.

User ID	Web Access Sequence	Frequent subsequence
100	*abdac*	*abac*
200	*eaebcac*	*abcac*
300	*babfaec*	*babac*
400	*afbacfc*	*abacc*

3 WAP-mine : Mining Access Patterns Efficiently from Web Logs

Access patterns can be mined using sequential pattern mining techniques. Almost all previously proposed methods for sequential pattern mining are based on a sequential pattern version of *Apriori heuristic* [1], stated as follows.

Property 1. **(Sequential Pattern Apriori)** Let $\mathcal{SEQ}$ be a sequence database, if a sequence G is not a ξ-pattern of $\mathcal{SEQ}$, any super-sequence of G cannot be a ξ-pattern of $\mathcal{SEQ}$.

For example, "f" is not a 75%-pattern of $\mathcal{WAS}$ in Example 1, thus any access sequence containing "f", cannot be a 75%-pattern.

The sequential pattern Apriori property may substantially reduce the size of candidate sets. However, because of the combinatorial nature of the sequential

pattern mining, it may still generate a huge set of candidate patterns, especially when the sequential pattern is long, which is exactly the case of Web access pattern mining.

This motivates us to study alternative structures and methods for Web access pattern mining. The key consideration is how to facilitate the tedious support counting and candidate generating operations in the mining procedure.

Our novel approach for mining Web access patterns is called WAP-mine . It is based on the following heuristic, which follows Property 1.

Property 2. **(Suffix heuristic)** If e is a frequent event in the set of prefixes of sequences in $\mathcal{WAS}$, w.r.t. pattern P, sequence eP is an access pattern of $\mathcal{WAS}$.

For example, b is a frequent event within the set of prefixes w.r.t. ac in Example 1, so we can claim that bac is an access pattern.

Basically, the general idea of our method can be summarized as follows.

- A nice data structure, WAP-tree , is devised to register access sequences and corresponding counts compactly, so that the tedious support counting can be avoided. It also maintains linkages for traversing prefixes with respect to the same suffix pattern efficiently. A WAP-tree registers all and only all information needed by the rest of mining. Once such a data structure is built, all the remaining mining processing is based on the WAP-tree . The original access sequence database is not needed any more. Because the size of WAP-tree is usually much smaller than that of the original access sequence database, as shown later, the mining becomes easier. As shown in Section 4, the construction of WAP-tree is quite efficient by simply scanning the access sequence database twice.
- An efficient recursive algorithm is proposed to enumerate access patterns from WAP-tree . No candidate generation is required in the mining procedure, and only the patterns with enough support will be under consideration. The philosophy of this mining algorithm is *conditional search.* Instead of searching patterns level-wise as Apriori , conditional search narrows the search space by looking for patterns with the same suffix, and count frequent events in the set of prefixes with respect to condition as suffix. Conditional search is a partition-based divide-and-conquer method instead of bottom-up generation of combinations. It avoids generating large candidate sets.

The essential structure of the WAP-mine algorithm is as follows. The algorithm scans the access sequence database twice. In the first pass, it determines the set of frequent events. An event is called a **frequent event** if and only if it appears in at least $(\xi \cdot |\mathcal{WAS}|)$ access sequences of $\mathcal{WAS}$, in which $|\mathcal{WAS}|$ and ξ denotes the number of access sequences in $\mathcal{WAS}$ and the support threshold, respectively. In the next scan, WAP-mine builds a tree data structure, called WAP-tree , using frequent events, to register all count information for further mining. Then, WAP-mine recursively mine the WAP-tree using *conditional search* to find all Web access patterns. An overview of the algorithm is given in Algorithm 1.

Algorithm 1 (WAP-mine : **mining access patterns in Web access sequence database**)
Input: access sequence database $\mathcal{WAS}$ and support threshold ξ $(0 < \xi \leq 1)$.
Output: the complete set of ξ-patterns in $\mathcal{WAS}$.
Method:

1. Scan $\mathcal{WAS}$ once, find all frequent events.
2. Scan $\mathcal{WAS}$ again, construct a WAP-tree over the set of frequent events for using Algorithm 2, presented in Section 4;
3. Recursively mine the WAP-tree using *conditional search*, which will be presented in Section 5.

There are two key techniques in our method, constructing WAP-tree and mining access patterns from WAP-tree . They are explored in detail in the following two sections. Section 4 focuses on the concept and the construction of WAP-tree , while Section 5 investigates the mining of access patterns from WAP-tree .

4 Construction of WAP-tree

The following observations may help us design a highly condensed Web access pattern tree.

1. Of all the 1-sequences, only the frequent ones will be useful in the construction of frequent k-sequences for any $k > 1$. Thus, if an event e is not in the set of frequent 1-sequences, there is no need to include e in the construction of a Web access pattern tree.
2. If two access sequences share a common prefix P, the prefix P can be shared in the WAP-tree . Such sharing can bring some advantages. It saves some space for storing access sequences and facilitates the support counting of any subsequence of the prefix P.

Based on the above observations, a **Web access pattern tree** structure, or WAP-tree in short, can be defined as follows.

1. Each node in a WAP-tree registers two pieces of information: label and count, denoted as *label* : *count*. The root of the tree is a special virtual node with an empty label and count 0. Every other node is labeled by an event in the event set E, and is associated with a *count* which registers the number of occurrences of the corresponding prefix ended with that event in the Web access sequence database.
2. The WAP-tree is constructed as follows: for each access sequence in the database, filter out any nonfrequent events, and then insert the resulting *frequent subsequence* into WAP-tree . The insertion of frequent subsequence is started from the root of WAP-tree . Considering the first event, denoted as e, increment the count of child node with label e by 1 if there exists one; otherwise create a child labeled by e and set the count to 1. Then, recursively insert the rest of the frequent subsequence to the subtree rooted at that child labeled e.

3. Auxiliary node linkage structures are constructed to assist node traversal in a WAP-tree as follows. All the nodes in the tree with the same label are linked by shared-label linkages into a queue, called **event-node queue**, The event-node queue of with label e_i is also called e_i**-queue**. There is one **header table** $\mathcal{H}$ for a WAP-tree , and the head of each event-node queue is registered in $\mathcal{H}$.

Example 2. Let's consider the access sequence database in Example 1. Suppose the support threshold is set to 75%, i.e. it is required to find all Web access patterns supported by at least three access sequences in the database.

One scan of the database derives the set of frequent 1-events: $\{a, b, c\}$. For convenience, the frequent subsequences are listed in the rightmost column of Table 1.

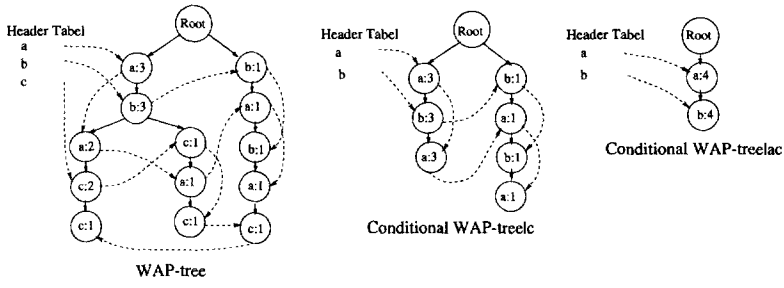


Fig. 1. The WAP-tree and conditional WAP-tree for frequent subsequences in Table 1.

The WAP-tree is shown in Figure 1, which is constructed as follows. First, insert the sequence *abac* into the initial tree with only one virtual root. It creates a new node $(a : 1)$ (i.e., labeled as a, with count set to 1) as the child of the root, and then derives the a-branch "$(a : 1) \rightarrow (b : 1) \rightarrow (a : 1) \rightarrow (c : 1)$", in which arrows point from parent nodes to children ones. Second, insert the second sequence *abcac*. It starts at the root. Since the root has a child labeled a, a's count is increased by 1, i.e., $(a : 2)$ now. Similarly, we have $(b : 2)$. The next event, c, does not match the existing node a, and a new child node $c : 1$ is created and inserted. The remaining sequence insertion process can be derived accordingly.

The algorithm for constructing a WAP-tree for Web access sequences is given in Algorithm 2.

Algorithm 2 (WAP-tree **Construction for Web access sequences**)
Input: A Web access sequence database $\mathcal{WAS}$ and the set of frequent events FE (which is obtained by scanning $\mathcal{WAS}$ once).
Output: an WAP-tree $\mathcal{T}$.

Method:

1. Create a root node for $\mathcal{T}$;
2. For each access sequence S in the access sequence database $\mathcal{WAS}$ do
   (a) Extract frequent subsequence S' from S by removing all events appearing in S but not in FE. Let $S' = s_1 s_2 \cdots s_n$, where s_i $(1 \leq i \leq n)$ are events in S'. Let *current_node* point to the root of $\mathcal{T}$.
   (b) For $i = 1$ to n do, if *current_node* has a child labeled s_i, increase the count of s_i by 1 and make *current_node* point to s_i, else create a new child node $(s_i : 1)$, make *current_node* point to the new node, and insert it into the s_i-queue.
3. Return($\mathcal{T}$);

Analysis: The WAP-tree registers all access sequence counts. As will be shown in later sections, the mining process for all Web access patterns needs to work on the WAP-tree only, instead of on the original database any more. Therefore, WAP-mine needs to scan the access sequence database only twice. It is easy to show that the height of the WAP-tree is one plus the maximum length of the frequent subsequences in the database. The width of the WAP-tree , i.e. the number of leaves of the tree, is bounded by the number of access sequences in the database. Therefore, WAP-tree may not generate explosive number of nodes. Access sequences with same prefix will share some upper part of path from root. Statistically, considering the factor of prefix sharing, the size of WAP-tree is much smaller than the size of access sequence database.

From Algorithm 2, the construction of WAP-tree , one can observe an important property of WAP-tree stated as follows.

Lemma 1. *For any access sequence in an access sequence database $\mathcal{WAS}$, there exists a unique path in the* WAP-tree *starting from the root such that all labels of nodes in the path in order is exactly the same as the events in the sequence.*

This lemma ensures that the number of distinct leaf nodes as well as paths in an WAP-tree cannot be more than the number of distinct frequent subsequences in the access sequence database, and the height of the WAP-tree is bounded by one (for the root) plus the maximal number of instances of frequent 1-events in an access sequence.

It is easy to show that a WAP-tree can be partitioned and structured in the form similar to B+-tree, and can be implemented even in pure SQL. Therefore, WAP-tree as well as mining using WAP-tree are highly scalable.

5 Mining Web Access Patterns from WAP-tree

The WAP-tree structure constructed by Algorithm 2 provides some interesting properties which facilitate mining Web access patterns.

Property 3. **(Node-link property)** For any frequent event e_i, all the frequent subsequences contain e_i can be visited by following the e_i-queue, starting from the record for e_i in the header table of WAP-tree .

The property facilitates the access of all the pattern information related to frequent event e_i by following the all branches in WAP-tree linked by e_i-queue only once. For any node labeled e_i in an WAP-tree , all nodes in the path from root of the tree to this node (excluded) form a **prefix sequence** of e_i, The count of this node labeled e_i is called the **count** of the prefix sequence.

Please note that a path from root may have more than one node labeled e_i, thus a prefix sequence of e_i may contain another prefix sequence of e_i. For example, sequence *aba* is a prefix sequence of "*b*" in *abab*, it contains another prefix sequence of "*b*", *a*. when counting *ab* in sequence *abab*, we must make sure no double counting, i.e. *abab* contributes only 1 to the count of *ab*. It is achieved by the concept of *unsubsumed count* as follows.

Let G and H be two prefix sequences of e_i, and G is also formed by the sub-path from root of that H is formed by, H is called a **super-prefix sequence** of G, and G is a **sub-prefix sequence** of H. For instance, *aba* is a super-prefix sequence of *a*.

For a prefix sequence of e_i without any super-prefix sequences, we define the **unsubsumed count** of that sequence as the count of it. For a prefix sequence of e_i with some super-prefix sequences, the *unsubsumed count* of it is the count of that sequence minus unsubsumed counts of all its super-prefix sequences. For example, let $S = (a : 6) \rightarrow (b : 5) \rightarrow (a : 2) \rightarrow (b : 2)$ be one path from root, the unsubsumed count of the first *a*, a prefix sequence of *b*, in the path should be 3 instead of 5, since two of the totally five counts in the first *b* node devotes to the super-prefix sequence *aba* of *a*.

Property 4. **(Prefix sequence unsubsumed count property)** The count of a sequence G ended with e_i is the sum of unsubsumed counts of all prefix sequences of e_i which is a super-sequence of G.

Based on the above two properties, we can apply **conditional search** to mine all Web access patterns using WAP-tree . What "conditional search" means, instead of searching *all* Web access patterns at a time, it turns to search Web access patterns with same **suffix**. Suffix is used as the condition to narrow the search space. As the suffix becomes longer, the remaining search space becomes smaller potentially.

The conditional search paradigm has some advantages against Apriori -like ones. The *node-link property* of WAP-tree ensures that, for any frequent event e_i, all sequences with suffix e_i can be visited efficiently using the e_i-queue of the tree. On the other hand, the *prefix sequence unsubsumed count property* makes sure that to count all frequent events in the set of sequences with same suffix, only caring the unsubsumed count is sufficient. That simplifies the counting operations. These two properties of WAP-tree make the conditional search efficient.

The basic structure of mining all Web access patterns in WAP-tree is as follows. If the WAP-tree has only one branch, all (ordered) combinations events in the branch are all the Web access patterns in the tree. So what needs to be done is just to return the complete set of such combinations. Otherwise, for each frequent event e_i in the WAP-tree , by following the e_i-queue started from header table, an e_i-**conditional access sequence base** is constructed, denoted

as $\mathcal{PS} \mid e_i$, which contains all and only all prefix sequences of e_i. Each prefix sequence in $\mathcal{PS} \mid e_i$ carries its count from the WAP-tree . For each prefix sequence of e_i with count c, when it is inserted into $\mathcal{PS} \mid e_i$, all of its sub-prefix sequences of e_i are inserted into $\mathcal{PS} \mid e_i$ with count $-c$. It is easy to show that by accumulating counts of prefix sequences, a prefix sequence in $\mathcal{PS} \mid e_i$ holds its unsubsumed count. Then, the complete set of Web access patterns which are prefix sequence of e_i can be mined by concatenating e_i to all Web access patterns returned from mining the conditional WAP-tree , and e_i itself.

The algorithm is given as follows.

Algorithm 3 (Mining all Web access patterns in a WAP-tree)
Input: a WAP-tree $\mathcal{T}$ and support threshold ξ.
Output: the complete set of ξ-patterns.
Method:

1. if the WAP-tree $\mathcal{T}$ has only one branch, return all the unique combinations of nodes in that branch.
2. initialize Web access pattern set $\mathcal{WAP} = \emptyset$. Every event in WAP-tree $\mathcal{T}$ itself is a Web access pattern, insert them into $\mathcal{WAP}$.
3. for each event e_i in WAP-tree $\mathcal{T}$,
   (a) construct a conditional sequence base of e_i, i.e. $\mathcal{PS} \mid e_i$, by following the e_i-queue, count conditional frequent events at the same time.
   (b) if the the set of conditional frequent events is not empty, build a conditional WAP-tree for e_i over $\mathcal{PS} \mid e_i$ using algorithm 2. Recursively mine the conditional WAP-tree
   (c) for each Web access pattern returned from mining the conditional WAP-tree , concatenate e_i to it and insert it into $\mathcal{WAP}$
4. return $\mathcal{WAP}$.

Example 3. Let us mine the Web access patterns in the WAP-tree in Figure 1. Suppose the support threshold is set to 75%. We start the conditional search from frequent event c. The conditional sequence base of c is listed as follows.

$$aba : 2, ab : 1, abca : 1, ab : -1, baba : 1, abac : 1, aba : -1$$

To be qualified as a conditional frequent event, one event must have count 3. Therefore, the conditional frequent events are $a(4)$ and $b(4)$. Then, a conditional WAP-tree , WAP-tree $\mid c$, is built, as also shown in Figure 1.

Recursively, the conditional sequence base of ca is built. It is $ab : 3, b : 1, ab : 1, b : -1$. The WAP-tree $\mid a$ is built, also shown in Figure 1. There is only one branch in the conditional tree, so all combinations are generated. Thus, the Web access patterns with suffix ac are $aac, bac, abac, ac$.

Then, we can construct the conditional sequence base for b in WAP-tree $\mid c$, and mine the conditional WAP-tree . The frequent patterns abc, bc can be found.

At this point, the conditional search of c is finished. We can use other frequent events in turn, to find all the other Web access patterns.

Following the properties presented ahead and considering the enumerating of access patterns in Algorithm 3, the correctness of WAP-mine can be shown.

Theorem 1. WAP-mine *returns the complete set of access patterns without redundancy.*

As can be seen in the example, and shown in our experiments, mining Web access patterns using WAP-tree has significant advantages. First, the WAP-tree is an effective data structure. It registers all count information for pattern mining, and frees the mining process from counting candidates by pattern matching. Secondly, the conditional search strategies narrow the search space efficiently, and make best uses of WAP-tree structure. It avoids the overwhelming problems of generating explosive candidates in Apriori -like algorithms.

6 Performance Evaluation and Conclusions

Experiments are pursued to compare the efficiency of WAP-mine and GSP, the algorithm proposed in [12]. The dataset for experiment is generated based on the principle introduced in [2]. All experiments are conducted on a 450-MHz Pentium PC machine with 64 megabytes main memory, running Microsoft Windows/NT. All the programs are written in Microsoft/Visual C++ 6.0.

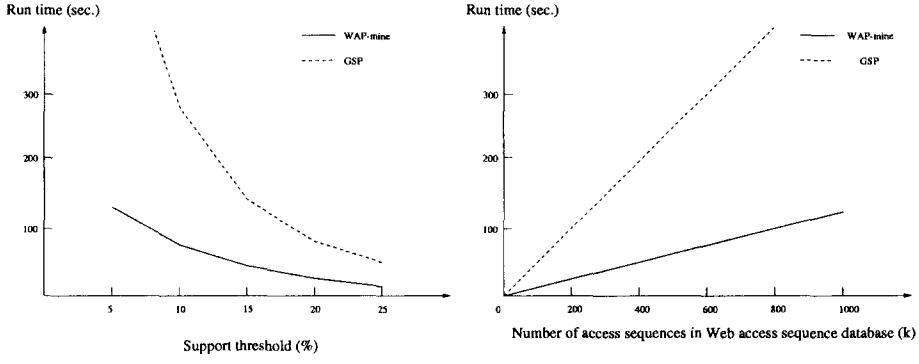


Fig. 2. Experimental results.

The experimental results are shown in Figure 2. We compare the scalabilities of our WAP-mine and GSP, with threshold as well as the number of access sequences in the database. The experimental result shows that WAP-mine outperforms GSP in quite significant margin, and WAP-mine has better scalability than GSP. Both WAP-mine and GSP show linear scalability with the number of access sequences in the database. However, WAP-mine outperforms GSP.

In conclusion, WAP-tree is an effective structure facilitating Web access pattern mining, and WAP-mine outperforms GSP based solution in a wide margin.

The success of WAP-tree and WAP-mine can be credited to the compact structure of WAP-tree and the novel *conditional search* strategies.

We believe that, with certain extensions, the methodology of WAP-tree and WAP-mine can be applied to attack many data mining tasks efficiently such as sequential pattern mining and episode mining.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994.
2. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 1995 Int. Conf. Data Engineering*, pages 3–14, Taipei, Taiwan, March 1995.
3. C. Bettini, X. Sean Wang, and S. Jajodia. Mining temporal relationships with multiple granularities in time sequences. *Data Engineering Bulletin*, 21:32–38, 1998.
4. R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining World Wide Web browsing patterns. In *Journal of Knowledge & Information Systems*, Vol.1, No.1, 1999.
5. J. Graham-Cumming. Hits and misses: A year watching the Web. In *Proc. 6th Int'l World Wide Web Conf.*, Santa Clara, California, April 1997.
6. J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *Proc. 1999 Int. Conf. Data Engineering (ICDE'99)*, pages 106–115, Sydney, Australia, April 1999.
7. H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. In *Proc. 1998 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98)*, pages 12:1–12:7, Seattle, Washington, June 1998.
8. H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.
9. B. Özden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proc. 1998 Int. Conf. Data Engineering (ICDE'98)*, pages 412–421, Orlando, FL, Feb. 1998.
10. M. Perkowitz and O. Etzioni. Adaptive sites: Automatically learning from user access patterns. In *Proc. 6th Int'l World Wide Web Conf.*, Santa Clara, California, April 1997.
11. M. Spiliopoulou and L. Faulstich. WUM: A tool for Web utilization analysis. In Proc. 6th Int'l Conf. on Extending Database Technology (EDBT'98), Valencia, Spain, March 1998.
12. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, pages 1–12, Montreal, Canada, June 1996.
13. T. Sullivan. Reading reader reaction: A proposal for inferential analysis of Web server log files. In *Proc. 3rd Conf. Human Factors & The Web*, Denver, Colorado, June 1997.
14. L. Tauscher and S. Greeberg. How people revisit Web pages: Empirical findings and implications for the design of history systems. In *Int'l Journal of Human Computer Studies, Special Issue on World Wide Web Usability*, 47:97-138, 1997.
15. O. Zaiane, M. Xin, and J. Han. Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs. In *Proc. Advances in Digital Libraries Conf. (ADL'98)*, Melbourne, Australia, pages 144-158, April 1998.

A Comparative Study of Classification Based Personal E-mail Filtering

Yanlei Diao, Hongjun Lu*, and Dekai Wu

Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{diaoyl,luhj,dekai}@cs.ust.hk

Abstract. This paper addresses personal E-mail filtering by casting it in the framework of text classification. Modeled as semi-structured documents, E-mail messages consist of a set of fields with predefined semantics and a number of variable length free-text fields. While most work on classification either concentrates on structured data or free text, the work in this paper deals with both of them. To perform classification, a naive Bayesian classifier was designed and implemented, and a decision tree based classifier was implemented. The design considerations and implementation issues are discussed. Using a relatively large amount of real personal E-mail data, a comprehensive comparative study was conducted using the two classifiers. The importance of different features is reported. Results of other issues related to building an effective personal E-mail classifier are presented and discussed. It is shown that both classifiers can perform filtering with reasonable accuracy. While the decision tree based classifier outperforms the Bayesian classifier when features and training size are selected optimally for both, a carefully designed naive Bayesian classifier is more robust.

1 Introduction

As the Internet grows at a phenomenal rate, electronic mail (abbreviated as E-mail) has become a widely used electronic form of communication on the Internet. Everyday, a huge number of people exchange messages in this fast and inexpensive way. With the excitement on electronic commerce growing, the usage of E-mail will increase more dramatically. However, the advantages of E-mail also make it overused by companies, organizations or people to promote products and spread information, which serves their own purposes. The mailbox of a user may often be crammed with E-mail messages some or even a large portion of which are not of interest to her/him. Searching for interesting messages everyday is becoming tedious and annoying. As a consequence, a personal E-mail filter is indeed needed.

The work on building an E-mail filter can be cast into the framework of text classification: An E-mail message is viewed as a document, and a judgement of

* The second author's work is partially supported by a grant from the National 973 project of China (No. G1998030414).

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 408-419, 2000.
© Springer-Verlag Berlin Heidelberg 2000

interesting or not is viewed as a class label given to the E-mail document. While text classification has been well explored and various techniques have been reported [2, 3, 7], empirical study on the document type of E-mail and the features of building an effective personal E-mail filter in the framework of text classification is only modest.

Along the line of empirical study on E-mail classification, Fredrik Kilander summarized real users' suggestions and opinions on what should be the important properties in classifying electronic texts and messages [4]. A preliminary study claimed that threading electronic mail [6] could only gain partial success based on structured information. A significant level of effectiveness could be achieved by applying standard text matching methods to the textual portions. A prototype, *Smokey* [12], combined natural language processing and sociolinguistic observations to identify insulting messages. This work differed from general electronic text classification, focusing mainly on language processing. A Bayesian approach to filtering junk E-mail was presented in [11]. It considered domain specific features in addition to raw text of E-mail messages. Elaborating on commercial junk E-mail, it enhanced the performance of a Bayesian classifier by handcrafting and incorporating many features indicative of junk E-mail. William Cohen compared a "traditional IR" method based on *TF-IDF* (*Term Frequency – Inverse Document Frequency*) weighting and a new method for learning sets of "keyword-spotting rules" based on the *RIPPER* rule learning algorithm [1]. The experiments, however, were only conducted with a relatively small number of data sets of real users. The issues related to building an effective E-mail classifier were not fully considered either.

The work reported in this paper was motivated by our belief that to realize an effective personal E-mail filter in the framework of text classification, the following issues should be fully taken into account.

- An E-mail filter is personalized and the knowledge used by each personal filter is subjective. Therefore, classifying personal E-mail messages is more challenging than using a priori knowledge to filter commercial junk messages that are often characterized by symbols and words like '$', "free", "saving", etc.
- An in depth study on the distinct type of E-mail documents is needed to make full use of the information embedded in them. Feature selection is the key issue.
- Typical text classification techniques should be examined and compared to enable better understanding of the capabilities and characteristics of these techniques to perform the task of a personal E-mail filter.
- A relatively large amount of real E-mail data from individuals with different interests should be used in experiments.

For the problem of classifying E-mail documents, the objects to be classified are semi-structured textual documents consisting of two portions. One portion is a set of structured fields with well-defined semantics and the other portion is a number of variable length sections of free text. We would like to emphasize this feature in our study because information from both portions is important. In the case of E-mail messages, the fields in the mail header such as the sender and the recipient are very informative when we determine how interesting the message part is. On the other hand, the interestingness of an E-mail message from the same sender also depends on the content of the body message. However, not many text classifiers take both portions into consideration. For example, the classic document clustering techniques in information retrieval seldom consider the contents of structured fields. On the other

hand, conventional classification techniques may not be effective when dealing with variable length free text.

There have been a number of approaches developed for classification. We selected two most popular approaches, naïve Bayesian classification [5, 8] and decision trees [9] to classify personal E-mail messages. The naïve Bayesian approach was chosen because it is widely used in text processing. Decision tree was chosen because of its effectiveness in classifying relational data. For the naïve Bayesian approach, a classifier based on previous work with some extensions was designed and implemented. For the decision tree approach, we implemented a classifier based on the widely used C4.5 system [10].

A series of experiments were conducted on a relatively large amount of real personal E-mail data. The behaviors of the two classification approaches were compared and discussed in detail. We find that both approaches provide reasonable performance in terms of recall rate and classification accuracy. Decision tree outperforms Bayesian a little when features and training size are selected optimally for both. However, the naïve Bayesian classifier is more robust with respect to the size and class disparity of training data.

The remainder of the paper is organized as follows. Section 2 discusses the modeling and features of E-mail messages. Section 3 presents our design and implementation of a naïve Bayesian classifier and a decision tree based classifier for E-mail filtering. The experiments and results are presented in Section 4. Finally Section 5 concludes the paper with discussions on future work.

2 Document Model

In this section, we describe how E-mail documents are modeled and how features are selected to perform personal E-mail filtering.

2.1 Modeling Semi-structured Documents

In a broad sense, E-mail messages are semi-structured documents that possess a set of structured fields with predefined semantics and a number of variable length free-text fields. In a formal way, such a document can be represented as Fig.1.

Field 1 to *Field s* are structured fields and usually contain information pertaining to the document, such as authorship, date, organization, layout of the text body, etc. As the major contents of the document, *Field s+1* to *Field s+t* are variable length free-text fields, such as subject area, abstract, the body and references. While most classification work focuses on either the structured part or the text part, we argue that both the structured fields and the free-text portion could contain important information for determining the class to which a document belongs. Therefore, a classifier to serve the purpose should be able to include features from both the structured fields and the free text.

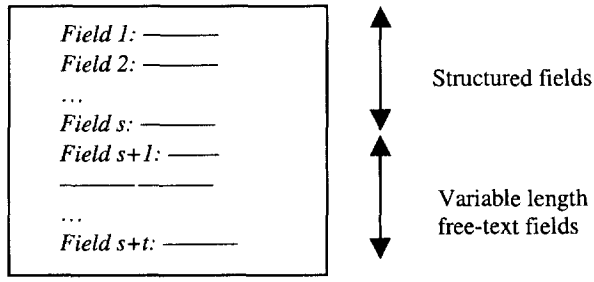


Fig. 1. Modeling Semi-structured Documents

2.2 Modeling Electronic Mail

In general, E-mail messages belong to the board class of semi-structured documents. Therefore they inherit the characteristics of possessing two portions of fields. In particular, they have some unique features. In addition to the structured fields and free text, there is evidence showing that domain specific information implicit in text fields is useful to improve the classification accuracy in certain applications. For example, Sahami et. al. reported that, there are many particular features of E-mail that help determine if a message is junk or not [11]. Such features include phases like "Free Money", and over-emphasized punctuation, such as "!!!". Since a feature in the free-text part normally refers to a single word, these particular features are treated as the third type, handcrafted features. To make full use of the information in an E-mail message, we generated all three types of features for each document.

- *Structured features*: features represented by structured fields in the header part of an E-mail document. In this work, six structured features were generated. They are *SenderDomain* (the domain of a sender, such as .com and .edu), *SenderAddress* (the E-mail address of a sender), *Recipient* (single recipient, in a group with the name mentioned, or via a mailing list), *Date*, *MailType* (replied, forwarded or sent directly), and *ContentType* (having attachment or not).
- *Textual features*: features explicitly contained in a free text section. In this work, only words consisting of alphabetic letters (no numbers or symbols) were counted into the vocabulary. Furthermore, a standard stop list was used to remove those words insignificant to classification. Simple stemming was also applied to reduce the vocabulary size.
- *Handcrafted features*: features obtained by preprocessing the documents. Heuristically six features were handcrafted. They are (1) the number of exclamation marks, (2) the number of dollar signs, (3) the number of http links, (4) the length of the message, (5) the length of the subject line, and (6) the occurrence of words indicative of not interesting E-mail in the subject area (a list of such words was collected in advance).

The usefulness of each type of features in different classifiers will be discussed in detail in experiments.

3 Two Classifiers

3.1 A Naïve Bayesian Classifier

The Bayesian learning approach views a text document as a bag of words. For a naive Bayesian classifier to work effectively, two conditions should be satisfied. First any word inside a document occurs independently. Second, there is no linear ordering of the word occurrences. Although these two assumptions may not hold in real cases, naïve Bayesian classifiers do provide good performance in a lot of applications [3, 8].

Two different generative models, the *multi-variate Bernoulli* model and the *multinomial* model under the Bayesian framework were reported in [8]. Experimental results show the multinomial model usually outperforms the multi-variate Bernoulli model when vocabulary size is large or chosen optimally for both. Thus, we adopted the multinomial model with a little simplification as shown from formula (1) to (4).

The following symbols are used in the rest part of this paper. $C_1 \ldots C_K$ are a set of class labels of a class variable C. $D_1 \ldots D_m$ are a set of training documents. $F_1 \ldots F_n$ represent a set of features in a given document. The class label of a document D' is determined as follows:

$$C = \arg\max_k P(C_k \mid D') = \arg\max_k P(D' \mid C_k) P(C_k). \tag{1}$$

Since a document is represented by a set of features $\{F_1 \ldots F_n\}$, with the naïve Bayes assumption that each feature in a document occurs independently, we have:

$$C = \arg\max_k P(F_1 \mid C_k) P(F_2 \mid C_k) \ldots P(F_n \mid C_k) P(C_k). \tag{2}$$

With a given set of labeled samples (the training data), the training process calculates *Bayes-optimal* estimates for all the parameters. Here the estimation of the probability of feature F_j on condition of class k and each class prior are obtained as follows:

$$P(F_j \mid C_k) = \frac{1 + \sum_{i=1}^{|D|} N(F_j, D_i) P(C_k \mid D_i)}{|V| + \sum_{t=1}^{|V|} \sum_{i=1}^{|D|} N(F_t, D_i) P(C_k \mid D_i)}, \tag{3}$$

$$P(C_k) = \frac{\sum_{t=1}^{|V|} \sum_{i=1}^{|D|} N(F_t, D_i) P(C_k \mid D_i)}{\sum_{k=1}^{K} \sum_{t=1}^{|V|} \sum_{i=1}^{|D|} N(F_t, D_i) P(C_k \mid D_i)}. \tag{4}$$

Here $N(F_j, D_i)$ is the number of occurrences of feature F_j in document D_i, $P(C_k \mid D_i) = \{0,1\}$ is given by the class label of that document, $|D|$ denotes the number of training documents and $\sum_{t=1}^{|V|} P(F_t \mid C_k) = 1$. To handle the probability of non-occurring features in the training data, add-by-one smoothing is used. $|V|$ is the vocabulary size.

Note that we are classifying E-mail messages that are distinct in document type. A feature involved in classification could be either a word in the text portion or a certain property (structured feature or handcrafted feature) associated to the document.

A Bayesian classifier has the advantage of being able to handle a large number of features. It simply models a document as "a bag of words" and all the words together form the vocabulary of the classifier. Naturally each word consisting of alphabetic letters in the main text portion is one feature in the vocabulary. To accommodate other two types of features in classification, a simple way is to treat such features as certain artificially created words and extend the vocabulary to include those features. The advantage of this approach is no need to modify the classifier. The importance of a feature is reflected uniformly by the probability of F_j on condition of class C_k no matter what type the feature belongs to.

Another issue of building a classifier in the context of E-mail messages is cost sensitiveness. When we assign a class label with the maximum class probability among all to a document, we are making an implicit assumption that the cost of misclassification is the same to all classes. In this application, the assumption is not true. Let C_1 denote the class label of "not interesting" and C_2 the class label of "interesting" (this notation will be used in the rest of the paper). The cost of misclassifying an interesting message to be not interesting is obviously much higher than that of misclassifying a not interesting message to be interesting. To make the naïve Bayesian classifier cost sensitive, we introduce to (2) one design parameter, threshold α_k for each class label k with $\sum_k \alpha_k = 1$:

$$C = \arg\max_k \left(\frac{P(F_1 \mid C_k)P(F_2 \mid C_k) \cdots P(F_n \mid C_k)P(C_k)}{\alpha_k} \right). \tag{5}$$

In this application with two class labels, the intuitive meaning of the threshold is as follows: In the case where misclassifying C_2 (interesting) into C_1 (not interesting) is more costly, we only make a prediction of class label C_1 if the final probability for decision making, $P(C_1|D')$, is greater than the threshold α_1, otherwise class label C_2 is chosen. In the rest part of the paper, for simplicity we use α to represent α_1. The classifier is cost sensitive with $\alpha > 0.5$. If we set $\alpha = 0.5$, we will have a normal cost-insensitive classifier.

3.2 A Decision Tree Based Classifier

Decision tree is a widely used data mining technique for classification and prediction, which is intensively studied in data mining applications in databases. C4.5, a typical and effective method of building decision trees, was used in our work to build a classifier of E-mail documents.

For a decision tree based approach, the situation is different from a Bayesian classifier. There is no problem for it to cope with the structured features and the handcrafted features since the number of these features (or attributes) is relatively small. However, it is not easy to handle a large number of textual features if every feature in the text is used in classification. A straightforward way is to limit the number of textual features that are considered by the classifier when a tree is built. In order to select textual features from the vocabulary, mutual information [8] is computed for each textual word F_i:

$$I(C; f_t) = \sum_{C_k \in C} \sum_{f_t \in \{0,1\}} P(C_k, f_t) \log(\frac{P(C_k, f_t)}{P(C_k)P(f_t)}) . \quad (6)$$

Here $f_t = 1$ indicates the presence of feature F_t in a document. $P(C_k)$ is the number of feature occurrences in documents with class label C_k divided by the total number of feature occurrences; $P(f_t)$ is the number of occurrences of feature F_t divided by the total number of feature occurrences; and $P(C_k, f_t)$ is the number of feature occurrences of F_t in documents with class label C_k divided by the total number of feature occurrences. Based on the $I(C; f_t)$ value a certain number of textual features are selected from the vocabulary as attributes that will be used in classification. For each document, the number of occurrences of a selected textual feature is the attribute value.

4 Experiments and Results

To have better understanding of the issues related to building a personal E-mail filter and the behavior of such filters, a series of experiments were conducted using both the naïve Bayesian classifier and the decision tree based classifier.

4.1 Data Sets and Performance Metrics

In the experiments, E-mail messages were used as document samples. The characteristics of collected data sets are shown in Table 1.

Table 1. Data Samples Used in Experiments

Source of data sets	5 (2 professors, 3 graduate students)
Number of data sets	11 (one set consists of E-mail messages in a month)
Size of each data set	250-700
Number of classes	2 ("not interesting", "interesting")

Every user who provided personal E-mail messages labeled all her/his messages as either interesting or not interesting. Since we did not give classification criteria to the person who provided the E-mail data, the classification was rather subjective. Unlike some other reported work, "not interesting" E-mail does not necessarily refer to commercial advertisements. For example, given two call-for-paper messages from international conferences in computer science, one may be classified as "not interesting" and the other as "interesting" depending on the theme of the conferences and the personal interest. Therefore, classifying an E-mail message as interesting or not is more challenging than pure commercial spam filtering.

During the experiments, each data set was divided into two portions: training data and test data in the chronicle order. The training data were used to train a classifier and the obtained classifier then classified the test data. Metrics used to measure the classification performance are defined as follows:

$$Error-rate = \frac{\#\ false\ classification}{\#\ classified\ messages}, \tag{7}$$

$$"Interesting"\ recall = \frac{\#"interesting"\ messages\ \ classified\ as\ "interesting"}{\#\ total\ "interesting"\ messages}, \tag{8}$$

$$"Interesting"\ precision = \frac{\#"interesting"\ messages\ classified\ as\ "interesting"}{\#\ total\ messages\ classified\ as\ "interesting"}. \tag{9}$$

"Not interesting" recall and precision are defined likewise. In the application of a personal E-mail filter, considering the information loss by misclassifying an "interesting" message as "not interesting", we emphasize the "interesting" recall and the error rate in the following tests.

4.2 Precision-Recall Graph of the Naïve Bayesian Classifier

The implemented Bayesian classifier classifies an E-mail message as "not interesting" only if the probability of "not interesting" is greater than threshold α ($\alpha \geq 0.5$). The value of the threshold in fact reflects the relative cost of misclassifying an "interesting" message as "not interesting". High α means high cost of such misclassification. Therefore, α is an important design parameter for a naïve Bayesian classifier. The first experiment aimed at the general behavior of the classifier when different threshold values were used. All three types of features were generated. By varying the threshold from 0.5 to 0.999, different recall and precision pairs were obtained for both "not interesting" and "interesting" classes. The average of 11 data sets was used to draw the recall-precision graph as shown in Fig. 2.

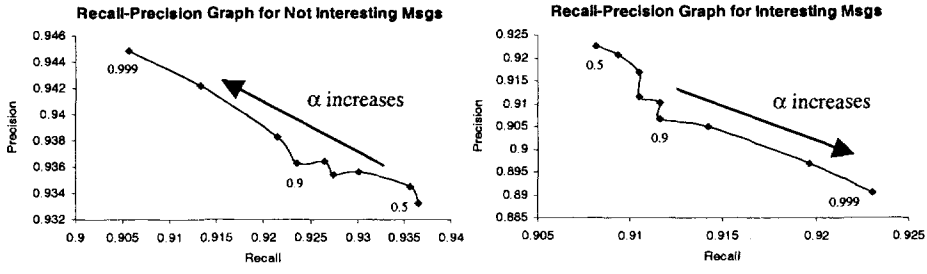


Fig. 2. Recall-Precision Graphs

A few observations can be made. First, within a wide range of the threshold value, both recall and precision values are around or above 90%. Second, the recall-precision curve of "not interesting" E-mail is better than that of "interesting" E-mail. Seemingly it is easier for a Bayesian classifier to identify "not interesting" E-mail

messages because they are often obviously characterized by some features. Finally, the rate at which recall and precision change is different from the rate at which threshold α changes. For "interesting" E-mail, when α increases from 0.5 to 0.9, the recall increases slowly by 0.35%. However when α increases from 0.9 to 0.999, the recall increases by 1.2%. Likewise "not interesting" recall decreases slowly as α changes from 0.5 to 0.9 but much faster when α changes from 0.9 to 0.999. Therefore, in order to obtain high recall of "interesting" E-mail α should be set a relatively high value, say higher than 0.9. In the following experiments, we used 0.99 as the default setting for α.

4.3 Experiments on Feature Selection

As we mentioned earlier, three types of features are generated for both classifiers. One question under exploration is how important these features are in E-mail classification. The second set of experiments was conducted to study the performance of the classifiers when different types of features were used. Fig. 3 and Fig. 4 depict the results of 11 data sets using the Bayesian classifier and the decision tree based classifier, respectively. In the figures, H stands for header features only, T for textual features only, HT for header features plus textual features, HH for header features plus handcrafted features, TH for textual features plus handcrafted, and HTH for header, textual and handcrafted features, namely all features. H, T, HT were three baselines. HH, TH, HTH were tested to detect the change of performance by adding handcrafted features to those baselines. The average of 11 groups was used for evaluation in terms of three accuracy metrics, error rate, "not interesting" recall and "interesting" recall.

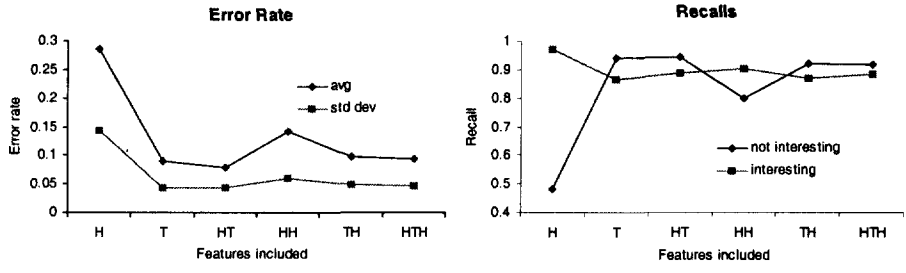


Fig. 3. Effects of Selected Features on the Naive Bayesian Classifier

Fig. 3 shows that, when only the structured features (H) are considered, the error rate is very high. If the structured features and the handcrafted features are selected (HH), the error rate is still quite high. However in these cases, the "interesting" recall is unexpectedly high. The reason lies in the small number of features involved in classification, only six or twelve. When only a small number of features in a document are selected, the difference between $P(D'|C_k)$ with different class label k is

outweighed by the denominator α. High α leads to a good "interesting" recall. However, error rate is high and "not interesting" recall is low, indicating these two feature selection methods are not appropriate to a Bayesian classifier. All other four cases involve the textual features. The best performance is achieved using the structured and the textual features (case HT). Adding header features better performs both case (T) and case (TH). However, comparing cases T and TH, HT and HTH, we find that, adding handcrafted features in fact does not improve the performance of case (T) and worsens that of (HT).

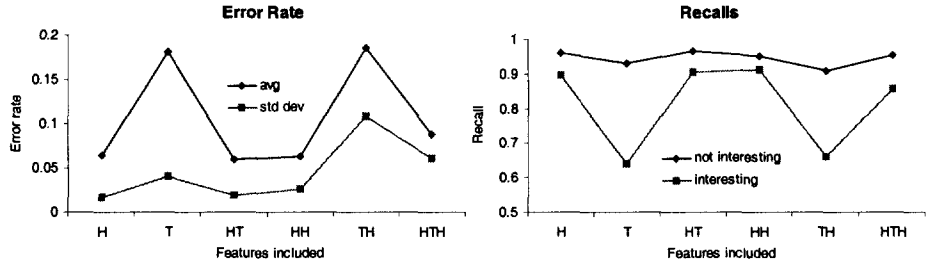


Fig. 4. Effects of Feature Selection on the Decision Tree Based Classifier

Fig. 4 shows the performance of the decision tree based classifier when different features are included in classification. Its performance is bad when the structured (header) features are not included. Therefore these two types of feature selection are not appropriate. On the baseline of H, adding either textual features or handcrafted features enhances the performance. However, when both textual features and handcrafted features are added to the baseline, the performance deteriorates, esp. in "interesting" recall and error rate. With all features included, the database schema consists of 32 attributes: 6 header features, 6 handcrafted features and 20 textual features. Decision tree becomes inaccurate with so many attributes. It works best with the selection method HT or HH.

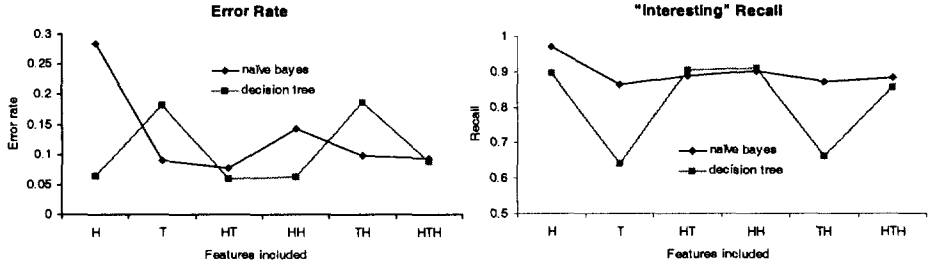


Fig. 5. Comparison in Feature Selection

Fig. 5 presents the average accuracy of the two classifiers in terms of error rate and "interesting" recall. Both the Naïve Bayesian classifier and the decision tree based classifier perform best with header features and textual features. The method of combining these two types of features for classification is useful. Neither of the classifiers works best with all features selected. One lesson we learned is that adding many features does not necessarily enhance the performance. Cautions should be taken in feature selection. In the optimal case, decision tree beats Bayesian based classifier in error rate and "interesting" recall.

4.4 Experiments on Robustness of Classifiers

We also conducted a set of experiments aiming to discover the robustness of both classifiers on different conditions that may happen in the real use of a personal E-mail filter. Limited by space, we just summarize the results without going into details.

Training size is an important issue that affects the accuracy of classifiers. From the experimental results we find when the training size is less than the test size, the decision tree based classifier has much lower "interesting" recall and higher error rate than the Bayesian classifier. It shows decision tree has a sparse data problem. As the training size grows, both classifiers improve the performance. In the optimal case decision tree outperforms naïve Bayesian. But a Bayes classifier keeps a reasonable performance on most conditions and has better performance when only a small training size is available.

Both classifiers can be affected by class disparity. Naïve Bayes classifier favors the major class by the factor of class prior in the decision rule. Decision tree based classifier chooses the major class at each test. Real users can have any ratio of "not interesting" messages to "interesting" messages. This experiment aimed to find out how these two classifiers perform as the class disparity of training data changes. The results show that the naïve Bayes classifier works well when "interesting" E-mail messages cover from 30% to 80% of the total training messages. The decision tree based classifier has high error rate at both ends of "interesting" coverage and the general performance is not stable.

5 Conclusion

This paper models E-mail messages as a combination of structured fields and free text fields, which motivated the work of classifying such documents deploying both kinds of information. Certain heuristic features obtained from preprocessing the documents were also included for the purpose of classifying E-mail messages. A comprehensive comparative study was conducted using a naïve Bayesian based classifier and a decision tree based classifier. Different ways of feature selection for both models were evaluated. Performance of two classifiers was compared with respect to training size and class disparity. By a series of experiments on real personal data, we find that both classifiers can be used to classify E-mail messages with reasonable accuracy. In the optimal cases, decision tree based classifier outperforms Bayesian classifier, but

Bayesian is more robust on various conditions. Careful feature selection from structured fields and free text body enhances performance.

The study reported in this paper can be extended in three directions. First, due to the personalized nature of electronic mail, the test data available is only moderately large. We are trying to collect more data from different types of users. It will deepen our study and enable more findings about how to achieve an effective personal E-mail filter. Second, we are exploring the ways of combining these two types of classifiers in feature selection and decision making, which might lead to a more accurate classification method in this problem domain. Last, we plan to expand the classification from two classes to multiple classes and further to a hierarchy of classes, which will better serve the need of E-mail users.

References

1. William W. Cohen: Learning Rules that Classify E-mail. In *Proceedings of the 1996 AAAI Spring Symposium on Machine Learning in Information Access*
2. W. W. Cohen, Y. Singer: Context-Sensitive Learning Methods for Text Categorization. In *Proceedings of SIGIR-1996*
3. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam and S. Slattery: Learning to Extract Symbolic Knowledge from the World Wide Web. In *Proceedings of the 15th National Conference on Artificial Intelligence* (AAAI-98)
4. Fredrik Kilander: Properties of Electronic Texts for Classification Purposes as Suggested by Users. http://www.dsv.su.se/~fk/if_Doc/F25/essays.ps.Z
5. D. D. Lewis: Naïve (Bayes) at Forty: The Independent Assumption in Information Retrieval. In *European Conference on Machine Learning*, 1998
6. D. D. Lewis, K. A. Knowles: Threading Electronic Mail: A Preliminary Study. In *Information Processing and Management*, 33(2): 209-217, 1997
7. D. D. Lewis, M. Ringuette: A Comparison of Two Learning Algorithms for Text Categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 81-93, Las Vegas, NV
8. Andrew McCallum and Kamal Nigam: A Comparison of Event Models for Naïve Bayes Text Classification. *Working notes of the 1998 AAAI/ICML workshop on Learning for Text Categorization*
9. J. R. Quinlan: Induction of Decision Trees. *Machine Learning*, 1: 81-106, 1986
10. J. R. Quinlan: *C4.5: Programs for Machine Learning*. San Mateo, Calif.: Morgan Kaufmann Publishers, 1993
11. M. Sahami, S. Dumais, D. Heckerman, E. Horvitz: A Bayesian Approach to Filtering Junk E-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*. AAAI Technical Report WS-98-05
12. Ellen Spertus: Smokey: Automatic Recognition of Hostile Messages. In *Proceedings of Innovative Applications of Artificial Intelligence* (IAAI) 1997

Extension of Graph-Based Induction for General Graph Structured Data

Takashi Matsuda, Tadashi Horiuchi, Hiroshi Motoda, and Takashi Washio

I.S.I.R., Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka 567-0047, JAPAN

Abstract. A machine learning technique called Graph-Based Induction (*GBI*) efficiently extracts typical patterns from directed graph data by stepwise pair expansion (pairwise chunking). In this paper, we expand the capability of the Graph-Based Induction to handle not only tree structured data but also multi-inputs/outputs nodes and loop structure (including a self-loop) which cannot be treated in the conventional way. The method is verified to work as expected using artificially generated data and we evaluated experimentally the computation time of the implemented program. We, further, show the effectiveness of our approach by applying it to two kinds of the real-world data: World Wide Web browsing data and DNA sequence data.

1 Introduction

Inductive learning, which tries to find useful rules and patterns from data, has been an important area of investigation. Conventional learning methods use an attribute-value table as a data representation language and represent the relation between attribute values and classes by use of decision tree [Quinlan86] and rules [Michalski90,Clark89]. Association rules [Agrawal94] widely used in the area of data mining belong to this type of data representation. However, the attribute-value table is not suitable for representing more general and structural data. Inductive logic programming (ILP) [Muggleton89] which uses the first-order predicate logic can represent general relationship in data. ILP has a merit that domain knowledge and acquired knowledge can be utilized as background knowledge. However, its state of the art is not so matured that anyone can use the technique easily.

By paying attention to the fact that many structural data involving relationship can be represented by a colored directed graph, we have proposed Graph-Based Induction (*GBI*) [Yoshida97] which can efficiently extracts typical patterns from a directed graph data of limited types by stepwise pair expansion called "pairwise chunking". The expressiveness of the *GBI* stands between the attribute-value table and the first-order predicate logic.

In this paper, we expand the capability of the *GBI* so that it can handle not only a tree structured data but also a graph data with multi-inputs/outputs nodes and loop structure (including a self-loop) which cannot be treated in the conventional way. The method is verified to work as expected using artificially

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 420–431, 2000.
© Springer-Verlag Berlin Heidelberg 2000

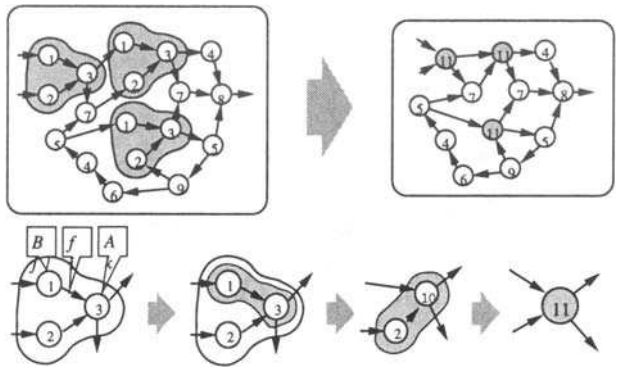


Fig. 1. The idea of graph contraction by pairwise chunking

generated data and we evaluated experimentally the computation time of the implemented program. We show the effectiveness of our approach by applying it to two kinds of the real scale data. One is an application to extracting typical patterns from WWW browsing histories data. Another is an application to extracting classification rules from two kinds of DNA sequence data.

2 Graph-Based Induction

2.1 Framework of Graph-Based Induction

The original *GBI* was so formulated to minimize the graph size by replacing each found pattern with one node that it repeatedly contracted the graph. The graph size definition reflected the sizes of extracted patterns as well as the size of contracted graph. This prevented the algorithm from continually contracting, which meant the graph never became a single node. Because finding a subgraph is known to be NP-hard, the ordering of links is constrained to be identical if the two subgraphs are to match, and an opportunistic beam search similar to genetic algorithm was used to arrive at suboptimal solutions. In this algorithm, the primitive operation at each step in the search was to find a good set of linked pair nodes to chunk (pairwise chunking) [Yoshida95].

The central intuition behind our *GBI* is as follows: a pattern that appears frequently enough in a colored directed graph is worth paying attention to and may represent an important concept in the environment (which is implicitly embedded in the input graph). In other words, the repeated patterns in the input graph represent typical characteristics of the given environment.

Because the search is local and stepwise, we can adopt an indirect index rather than a direct estimate of the graph size to find the promising pairs. On the basis of this notion, we generalize the original *GBI*. The idea of pairwise chunking is given in Figure 1.

The stepwise pair expansion (pairwise chunking) is performed by repeating the following three steps.

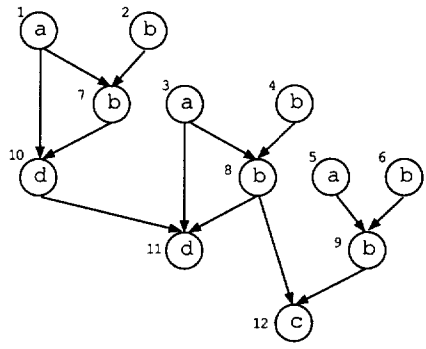


Fig. 2. An example of directed graph

Step 1. If there are patterns which are the same as the chunked pattern in the input graph, rewrite each of them to one node of the same label.
Step 2. Extract all pairs which consist of the connected two nodes in the graph.
Step 3. Select the most *typical* pair among the extracted pairs and register it as the pattern to chunk.

Stepwise pair expansion is performed by repeating the above three steps from the initial state where no typical pattern is yet found. As a result, the characteristics in data can be extracted as a set of typical patterns.

2.2 Graph-Based Induction for General Graph Structured Data

In order to apply *GBI* to general graph structured data, we adopt a method to represent the graph structured data using a set of table forms by paying attention to the link information between nodes. More concretely, the directed graph as shown in Figure 2 can be represented using Table 1. For example, the first line in this table shows that the node No.1 has node name "a" and also has nodes No.7 and No.10 as child nodes. In this way, directed graph data can be represented using the table forms. Further, the restriction of the link ordering is no more required.

As we count the pair (parent node $\rightarrow$ child node), it is necessary to identify self-loop when the parent node and the child node are of the same kind (*E.g.* $a \rightarrow a$). Therefore, we introduce "self-loop distinction flag".

Moreover, each time we perform the pairwise chunking, we keep link information between nodes in order to be able to restore the chunked pairs to the original patterns. This is realized by keeping two kinds of node information. One is "child node information" that means which node in the pattern the link goes to, and another is the "parent node information" that means which node in the pattern the link comes from. These two kinds of information are also represented by tables (not shown here). Chunking operation can be handled by manipulating these three tables.

Table 1. An example of table form translated from the directed graph

Node No.	Node Name	Child Node No.
1	a	@7@10
2	b	@7
3	d	@8@11
4	b	@8
5	a	@9
6	b	@9
7	d	@10
8	b	@11@12
9	b	@12
10	a	@11
11	b	
12	c	

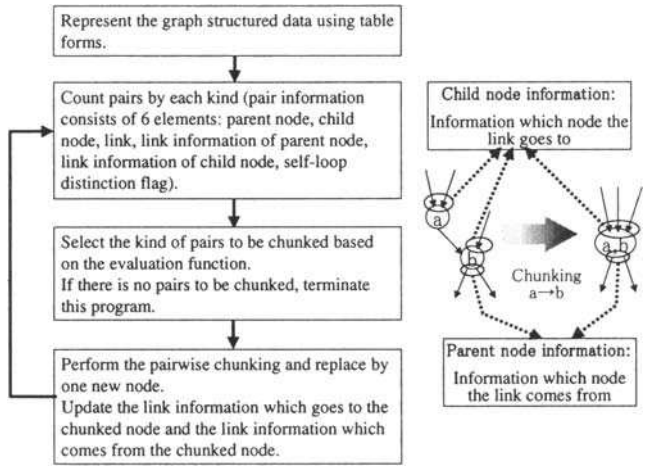


Fig. 3. Algorithm of the proposed method

The basic algorithm of the proposed method which extends *GBI* to handle a general graph structured data is shown in Figure 3. In this implemented program, we use the simple "frequency" of pairs as the evaluation function to use for stepwise pair expansion.

3 Performance Evaluation

The method was verified to work as expected using artificially generated data and we evaluated experimentally the computation time of the implemented program. The computation time is measured for 30,000 times repetition of program execution excluding the initialization steps.

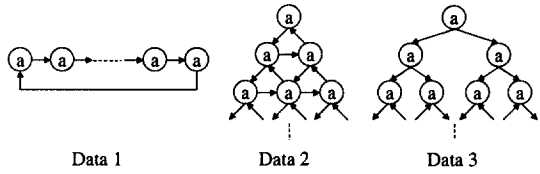


Fig. 4. Graph structured data for evaluation (Data 1CData 2CData 3)

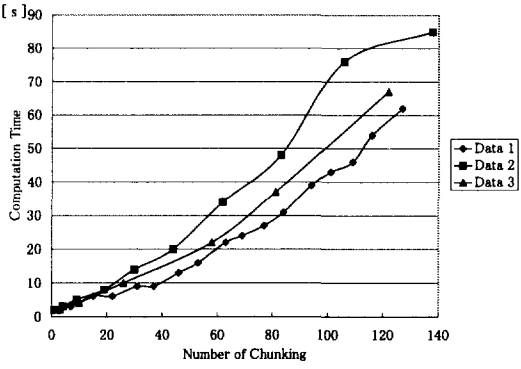


Fig. 5. Computation time and number of chunking (1)

3.1 Computation Time for Uncolored Graphs

At first, we evaluated the computation time using three kinds of graph structured data (Data 1: loop type, Data 2: lattice type, Data 3: tree type) as shown in Figure 4 for which there is only one kind of node label.

The computation time for chunking in the implemented program was measured as we increased the graph size from a small graph with several nodes to the graph which needs chunking about 120 times. Figure 5 shows the relationship between the computation time and the number of chunking.

From this figure, it is found that the computation time increases almost linearly with the number of chunking. And also it is considered that the gradient of each line depends on the average number of links going out from each node.

3.2 Computation Time for Colored Graphs

Next, we evaluated the computation time using three kinds of graph structured data (Data 4: loop type, Data 5: lattice type, Data 6: tree type) as shown in Figure 6 for which there are three kinds of node labels.

The computation time was measured in a similar way for uncolored graphs. Figure 7 shows the relationship between the computation time and the number of chunking. Overall, tendency is the same for uncolored graph. Compared with Figure 5, it is conjectured that the number of node labels does not affect the computation time so much.

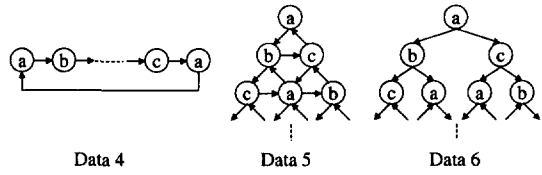


Fig. 6. Graph structured data for evaluation (Data 4CData 5CData 6)

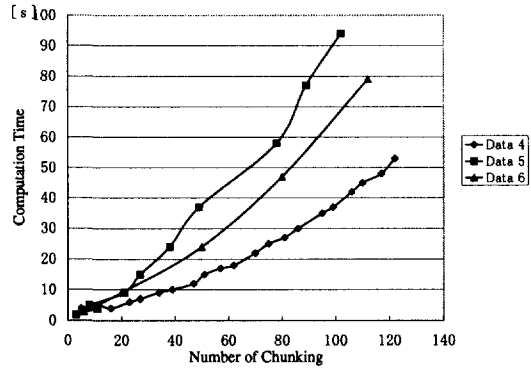


Fig. 7. Computation time and number of chunking (2)

We further confirmed this tendency for both uncolored and colored graphs by measuring the computation time for chunking as we increase the graph size from 50 nodes to 1000 nodes, where graphs are artificially generated in such a way that the average number of links going out of each node remains a fixed value.

4 Extracting Typical Patterns from WWW Browsing History Data

4.1 WWW Browsing History Data

The performance of the proposed method was examined through a real scale application in this section. The data analyzed is the log file of the commercial WWW server of Recruit Co., Ltd. in Japan. The URLs on WWW form a huge graph, where each URL represents a node that is connected by many links (other URLs). When a client visits the commercial WWW site, he/she browses only a small part of the huge graph in one access session, and the browsing history of the session becomes a small graph structured data. This site's total number of hit by the nation wide internet users always remains within the third place from the top in every month in Japanese internet record.

Table 2. Experimental result

Thresholdi%j	0.10	0.05	0.025
No. of Extracted Patterns	33	106	278
No. of Pairwise Chunking	9455	17443	26672
Computation Timei*Sec.*j	1374	1734	2187

4.2 Experimental Method

The basic format of an access record to a URL by a client in the log file is indicated in Figure 8. This access log of the WWW server for a particular day was over $400MB$ and the number of clients who access to the WWW server was about 26,800 on that day. The total number of the URLs involved in this commercial WWW site is about 19,400 and there are a large number of links between them.

As the log file consists of the sequence of the access records, they are initially sorted by the IP addresses, and each subsequence having an identical IP address corresponding to the browsing access history in a session of an individual client is extracted. Then, we transformed the subsequence of each client's visiting history into a graph structured data (total 150,000 nodes). By using all subgraphs transformed in this way as the input data, the proposed method extracted typical patterns in the data. In other words, after removing all kinds of error such as a server error from the access records and sorting the data in order of IP address, we make graph structured data for each IP address (client), and append them into a large table.

In the implemented program, we use the simple "frequency" of pairs as the evaluation function for stepwise pair expansion. We terminated the chunking when we finish finding all chunk patterns which consist of more than a certain number of nodes. We use the frequency threshold 0.1%C0.05%C0.025% of total nodes in the graph.

4.3 Experimental Results

We executed the implemented program using the experimental method described in the previous section. Table 2 shows the number of extracted chunk patterns, the number of pairwise chunking and the computation time for each frequency threshold.

Figure 9 indicates the relationship between the threshold and the number of derived patterns consisting of more than 3 nodes. The number of nodes in chunk patterns increases with the decrease of threshold because the larger chunk

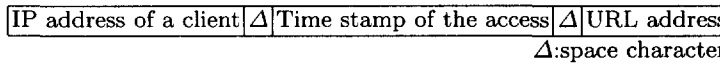


Fig. 8. Basic format of an access record.

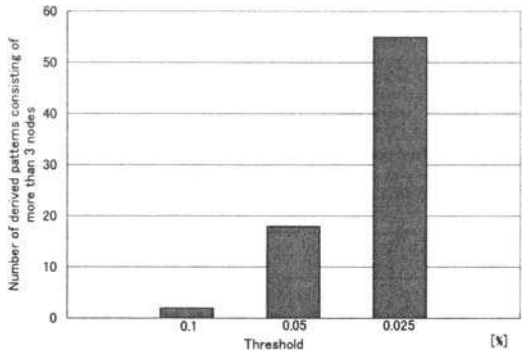


Fig. 9. Relationship between the threshold and the number of derived patterns consisting of more than 3 nodes

chunk patterns increases with the decrease of threshold because the larger chunk patterns are derived for the lower threshold where additional nodes are appended to the chunk patterns which have been already extracted in the higher threshold. Chunk patterns derived in the higher threshold is a subset of chunk patterns extracted in the lower threshold.

Several examples of the extracted chunk patterns are shown below.

$a)$ $/NAVI/CATEGORY/Sub/s03/ss13/d111.html$
$\rightarrow /NAVI/CATEGORY/Sub/s03/ss13/d112.html$
$\rightarrow /NAVI/CATEGORY/Sub/s03/ss13/d113.html$
$b)$ $/NAVI/CATEGORY/Sub/s01.html$
$\rightarrow /NAVI/CATEGORY/Sub/s01/ss06.html$
$\rightarrow /NAVI/CATEGORY/Sub/s01/ss06/d07.html$
$c)$ $/NAVI/mscategory/Sub/s12.html$
$\rightarrow /NAVI/mscategory/Sub/s08.html$
$\rightarrow /NAVI/mscategory/Sub/s02.html$
$\rightarrow /service/shank/NAVI/COOL/cool2.html$

Browsing pattern $a)$ is an example that clients follow some URLs in the same directory and the number of this pattern was 152 (*i.e.*, 152 clients followed this pattern). Browsing pattern $b)$ is an example that clients go deeper into directories step by step and the number of this pattern was 144. Browsing pattern $c)$ is an example that clients jump to the URLs in a different directory after following some URLs in the same directory and the number of this pattern was 72.

It is natural that many patterns similar to $a)$ or $b)$ have been extracted because of the structure of this WWW site. However, it is more interesting to note that some patterns such as $c)$ also have been extracted if the contents of each URL were available to us.

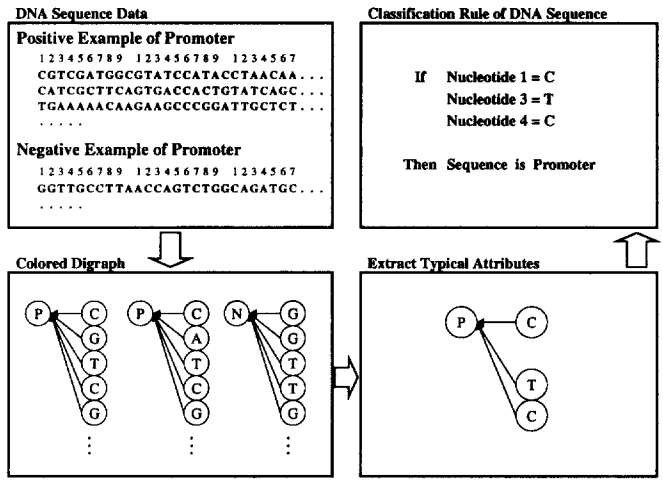


Fig. 10. Extraction of classification rules from DNA sequence data

5 Extracting Classification Rules from DNA Data

5.1 Application to Promoter DNA Sequence Data

In this section, the real-world task of recognizing biological concepts in DNA sequences is investigated. In particular, the task is to recognize *promoters* in strings that represent nucleotides (one of A, G, T, or C). A promoter is a genetic region which initiates the first step in the expression of an adjacent gene (*transcription*). This data set is one of the UCI Machine Learning Repository [Blake98]. The input features are 57 sequential DNA nucleotides and the total number of instances is 106 including 53 positive instances (sample promoter sequences) and 53 negative instances (non-promoter sequences).

Figure 10 illustrates the process of mapping the problem into a colored directed graph, using *GBI* method to extract patterns and interpreting them as classification rules. In mapping the cases in the data set into the graph structure, we construct one subgraph for each sequence in the data set. The subgraph consists of a root node and a set of leaf nodes. The color of the root node of the subgraph specifies whether the corresponding sequence represents a promoter sequence or not, which means the class information (positive or negative). The color of the i-th leaf specifies the nucleotide (one of A, G, T, or C).

In case of the classification problem, we interpret the root node as a class node and the links attached to it as the primary attributes. The node at the other end of each link is the value of the attribute, which can have secondary attributes, although this is not the case for this simple DNA problem. Thus, each attribute can have its own attributes recursively, and the graph (*i.e.*, each instance of the

Table 3. Experimental results

Learning Method	ID3	C4.5	*GBI*
No. of Errors /106	19	18	16

data) becomes a directed tree. Here, we have to select the attribute and its value pair that best characterizes the class.

The chunk patterns derived by *GBI* are tried to match for the test cases in the following way. The chunk patterns which have lower evaluation function value (frequency) are tried to match first. If the frequency of the chunk patterns is same, those which have more nodes in the pattern are tried to match first. That is, more specific rules are tried to match with higher priority.

Table 3 shows the experimental results (number of errors in total 106 cases) in comparison with other learning methods such as ID3, C4.5, which are evaluated by leaving-one-out. From this table, it is found that the error rate of *GBI* is lower than the standard tree-induction program ID3 and C4.5.

5.2 Application to Splice DNA Sequence Data

Splice junctions are the points on DNA sequence at which "superfluous" DNA is removed during the process of protein creation. The problem is to recognize the boundaries between *exons* (the parts of the DNA sequence retained after splicing) and *introns* (the parts of the DNA sequence that are spliced out) in a given sequence of DNA. This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as *E/I*), and recognizing intron/exon boundaries (referred to as *I/E*).

This data set contains 3190 cases, of which 25% are *I/E*, 25% are *E/I* and the remaining 50% are *Neither*. Each example consists of a 60 nucleotide DNA sequence categorized according to the type of boundary at the center of the sequence.

In mapping the cases in the data set into the graph structure, we constructed one subgraph for each sequence in the data set, in the same way as in the Promoter DNA data. The subgraph consists of a root node and a set of leaf nodes. The color of the root node of the subgraph specifies whether the corresponding sequence represents one of the *I/E*, *E/I* and *Neither*. The color of the i-th leaf specifies the nucleotide (one of A, G, T, or C).

The chunk patterns derived by *GBI* are tried to match for the test cases in the same way as mentioned in the previous subsection.

Table 4 shows the experimental results (error rate) in comparison with other learning methods such as ID3, C4.5, which are evaluated by 10-fold cross--validation. From this table, it is found that the error rate of *GBI* is lower than ID3 and is almost the same as the standard tree-induction program C4.5.

Table 4. Experimental results

Learning Method	ID3	C4.5	*GBI*
Error Rate (%)	10.6	8.0	8.8

6 Related Work

Most of the current methods for extracting knowledge from databases have difficulties in handling the growing amount of structural data that express relationships among data objects. However, there are some research work for discovering knowledge in structural data, especially graph structured data.

[Cook94] proposed the substructure discovery system called SUBDUE which discovers interesting and repetitive subgraphs in a labeled graph representation. Experiments show SUBDUE's applicability to several domains, such as molecular biology, image analysis and computer-aided design. SUBDUE expands one subgraph based on the Minimum Description Length (MDL) principle. Therefore, the number of substructure which is discovered in a graph is always one. On the other hand, *GBI* for general graph structured data which is proposed in this paper can extract multiple patterns based on the evaluation function for stepwise pair expansion.

[Wallace96] presented a Bayesian approach to the discovery of causal models based on Minimum Message Length (MML), which is one way to realize Occam's razor just like MDL. The MML induction approach can recover causal models from sample graph data without incorporating background knowledge. While this approach is towards automated learning of causal model using MML, this applicability to huge graph structured data is not clear so far.

[Inokuchi99] applied Basket Analysis to mine association rules from the graph structured data. The Basket Analysis [Agrawal94] derives frequent itemsets and association rules having *support* and *confidence* levels greater than their thresholds from massive transaction data. In [Inokuchi99], a simple preprocessing of data enabled to use a standard Basket Analysis technique for a graph structured data. However, each node must have a distinct label with this approach.

7 Conclusion

In this paper, we showed how we can expand the capability of the Graph-Based Induction algorithm to handle more general graphs, i.e., directed graphs with 1) multiple inputs/outputs nodes and 2) loop structure (including a self-loop). The algorithm was implemented and verified to work as expected using first artificially generated data and second two kinds of real-world data: World Wide Web browsing data and DNA sequence data. The algorithm runs almost linearly to the graph size and can indeed find interesting patterns.

The followings are in progress: 1) investigating the sensitivity of chunk ordering, 2) using extended chunks as constructed of new features for standard decision tree algorithm, 3) using statistical index (e.g. Gini Index [Breiman84]) or the description length in stead of the simple "frequency" as the evaluation function for stepwise expansion, 4) introducing a new index which corresponds to the notion of "similarity" of human concept, 5) applying different kinds of graph structured data in the real-world such as chemical structured data.

Acknowledgments

The authors are very grateful to Mr. Kouhei Kumazawa and Mr. Shoei Arai in Recruit Co., Ltd., Japan for providing the access log of the commercial WWW site.

References

[Agrawal94] R. Agrwal and R. Srikant. First Algorithms for Mining Association Rules. *Proc. of the 20th VLDB Conference,* pp. 487–499, 1994.

[Blake98] C. Blake, E. Keogh, and C. J. Merz. UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.

[Breiman84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[Clark89] P. Clark and T. Niblett. The CN2 Induction Algorithm. *Machine Learning* Vol. 3, pp. 261–283, 1989.

[Cook94] D. J. Cook and L. B. Holder. Substructure Discovery Using Minimum Description Length and Background Knowledge. *Journal of Artificial Intelligence Research*, Vol. 1, pp. 231–255, 1994.

[Inokuchi99] A. Inokuchi, T. Washio and H. Motoda. Basket Analysis for Graph Structured Data, *Proc. of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'99),* pp. 420–431, 1999.

[Michalski90] R. S. Michalski. Learning Flexible Concepts: Fundamental Ideas and a Method Based on Two-Tiered Representaion. *In Machine Learning, An Artificial Intelligence Approach*, Vol. III, (eds. Kodrtoff Y. and Michalski T.), pp. 63–102, 1990.

[Muggleton89] S. Muggleton and L. de Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming* Vol. 19, No. 20, pp. 629–679, 1994.

[Quinlan86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, Vol. 1, pp. 81–106, 1986.

[Wallace96] C. Wallace, K. B. Korb and H. Dai. Causal Discovery via MML, *Proc. of the 13th International Conference on Machine Learning (ICML'96)*, pp. 516–524, 1996.

[Yoshida95] K. Yoshida and H. Motoda. CLIP: Concept Learning from Inference Pattern, *Artificial Intelligence*, Vol. 75, No. 1, pp. 63–92, 1995.

[Yoshida97] K. Yoshida and H. Motoda. Inductive Inference by Stepwise Pair Extension (in Japanese), *Journal of Japanese Society for Artificial Intelligence*, Vol. 12, No. 1, pp. 58–67, 1997.

Text-Source Discovery and *GlOSS* Update in a Dynamic Web

Chi-Yuen Ng, Ben Kao, and David Cheung

Department of Computer Science and Information Systems,
The University of Hong Kong, Pokfulam, Hong Kong.
{cyng,kao,dcheung}@csis.hku.hk

Abstract. "Text-source discovery" is the problem of identifying relevant document databases that potentially contain documents that match a user query. *GlOSS* [6] is a cost-effective technique for solving the text-source discovery problem. However, the *GlOSS* approach assumes that the document databases are fully cooperative in exporting statistical information about their collections. This paper discusses how the *GlOSS* technique can be applied to a *dynamic* and *uncooperative* Web environment in assisting users to locate relevant Web information sources.
keywords: text-source discovery, *GlOSS*, search engines

1 Introduction

The World Wide Web enables its users to access large amounts of interesting information made available by various information sources. While new information is being put onto the Web everyday, large numbers of articles are being updated regularly, some at a very high frequency. It is impossible for a human being to keep track of all this information and changes. In order to fully utilize the power of the Web as a gigantic information source, it is essential to develop software systems on top of the Web to assist users in retrieving relevant documents.

Internet search engines such as Alta Vista, Lycos, etc., are the most popular tools that people use to locate information on the Web. A search engine works by traversing the Web via the pages it has encountered, and indexing the pages based on the keywords they contain [1].

Although search engines have been proven in practical use as indispensable tools for Web retrieval, they suffer from a number of drawbacks [2,3,4,5]. For many queries, the result is a very large answer set of *individual* Web pages with poor retrieval precision. As an example, if one queries some general information about "Microsoft Windows", up to thousands of individual pages that contain "Microsoft Windows" will be suggested. A more satisfactory answer to such a *general concept* query might be a recommendation to the Microsoft Web site. Starting from the main index page, the user can search via the query interface or navigate the Web site via the hypertext links according to his own specific interest to pin-point relevant documents. We call this strategy of first locating a relevant Web site, then locating relevant information within that site, the

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 432–441, 2000.
© Springer-Verlag Berlin Heidelberg 2000

two-step approach. In contrast, we consider the traditional search engines take a *one-step approach* to retrieve all potential pages directly.

As the Web develops, we see more and more information sources that are dedicated to specific topics of interests. If a user is looking for general information about a topic, chances are that a site exists on the Web that is specialized on that topic. Recommending Web sites instead of individual pages becomes more meaningful to this type of general concept queries.

One economical solution to recommend a Web site is to borrow the idea of *GlOSS*[6,7]. It is a system that, given a user query, recommends relevant document databases based on some statistical information that it keeps about the document databases. Such a problem of identifying relevant text databases is called the *text-source discovery problem.* The original study on *GlOSS* assumes that the document databases are fully cooperative in exporting statistical information about their document collections. However, this is obviously not the case for Web sites. The goal of this paper is to investigate how the idea of *GlOSS* can be applied in a *dynamic* and *uncooperative* Web environment.

The rest of the paper is organized as follows. In Section 2 we discuss some techniques for the text-source discovery problem. In Section 3 we propose a solution to the *GlOSS* update problem. In Section 4 we present experimental results verifying the effectiveness of our solution. Finally, we conclude the paper in Section 5.

2 Text-Source Discovery

GlOSS is one of the techniques tackling the text-source discovery problem. The sources are ranked according to the number of *relevant* documents they contain. Relevancy here is measured by the *similarity* between a document and the user query. There are two models in *GlOSS* to measure the similarity: the *Boolean* model and the *Vector-Space* model. In this paper, we focus our discussion on the *Boolean* model.

Under the Boolean model, a document, D is relevant to a query, Q if D contains all the keywords in Q. The similarity function, $sim(Q, D)$ is simply defined as:

$$sim(Q, D) = \begin{cases} 1 \text{ if } D \text{ contains all the keywords in } Q; \\ 0 \text{ otherwise.} \end{cases}$$

The *goodness* of a document database, *db*, with respect to Q, is measured by simply counting the number of matching documents:

$$goodness(Q, db) = \sum_{D \in db} sim(Q, D)$$

All the quantities defined above can be easily calculated with a full-text index as in the case of traditional search engines. However, if compared with *GlOSS*, the search engine approach is very costly, both in terms of the storage as well as

the network bandwidth requirements. This is because a search engine essentially has to keep the details of all the Web pages and reload them if they are updated.

In this paper, we apply the idea of *GlOSS* on the Web. The ultimate goal is to recommend Web sites based on the above word statistics. For each Web site, *db*, *GlOSS* keeps the total number of documents, $n(db)$, and the document frequency, $d_j(db)$, for each keyword j. The keywords are assumed to appear in the different documents independently and uniformly.[1] Given a conjunctive query Q of k words $t_{i_1}, ..., t_{i_k}$, *GlOSS* estimates the goodness of *db* with respect to Q by:

$$goodness(Q, db) = \prod_{j=1}^{k} \frac{d_{i_j}(db)}{n(db)} \times n(db) = \frac{\prod_{j=1}^{k} d_{i_j}(db)}{(n(db))^{k-1}} \quad (1)$$

GlOSS assumes the availability of the above keyword statistics. However, in practice, most Web sites are dynamic and uncooperative, i.e., they do not actively export the keyword statistics of their collections. Even if such statistics are obtained, they become stale fast due to frequent updates on the Web sites' contents. Maintaining these statistics is thus a very critical problem in applying *GlOSS* to tackle the Web site discovery problem.

3 *GlOSS* Update

In this section we demonstrate how the total number of documents, $n(db)$, and the document frequency $d_j(db)$'s are maintained, and how they are used to answer query under the Boolean model. Also, we describe the design of a prototype system which implements *GlOSS* in the Web environment.

It is not always possible to obtain the complete document collection of a Web site. Retrieving all documents poses much burden on the network bandwidth and storage. Also, in many cases, obtaining the complete collection is not necessary if the goal of the system is to determine whether a Web site contains a *non-trivial* number of relevant documents. As an example, the sports site ESPN is a good source of NBA basketball and contains numerous articles on the topic. To deduce that ESPN is a site relevant to the keyword "NBA", it is sufficient to examine only a fraction of articles located at ESPN. Hence, instead of *actively* probing a Web site for documents, we take a *passive* approach: our modified *GlOSS* server only examines and summarizes documents that a user community has ever retrieved from the Web site within a certain period of time. That means our system uses a *sample* of documents to project the information content of the Web site. Figure 1 shows the architecture of our system.

We assume that users access the Web via a Proxy server which buffers recently accessed Web documents in the cache. The Proxy cache is assumed to be large enough to store the documents retrieved in a day. The HTTP header of

[1] These assumptions are not realistic, and thus the estimates of *goodness* are off. However, the estimates are only used to *rank* document databases, not to compute accurate values. Readers are referred to [6] for a discussion on the efficacy of *GlOSS*.

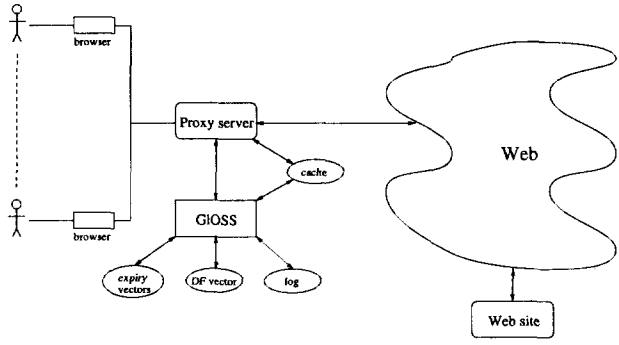


Fig. 1. System architecture

the document retrieved carries the last-modified time from which the Proxy can determine whether the document exists in the cache and whether it is updated.

Our *GlOSS* server interacts closely with the Proxy server. For each Web site, *db*, the *GlOSS* server maintains a *document frequency vector*, $DF(db)$. It models the document frequency statistics of the Web documents that our system has ever retrieved from *db* in a certain period of time, say, in the past M days. If $\tilde{db}$ denotes a subset of documents retrieved from *db*, then the j-th component of the $DF(db)$ vector represents the document frequency of the j-th keyword in $\tilde{db}$, denoted by $d_j(\tilde{db})$. Besides $DF(db)$, the *GlOSS* server also keeps the total number of documents in $\tilde{db}$, denoted by $n(\tilde{db})$. The goodness of *db* with respect to a query Q is estimated based on Equation 1 using $d_j(\tilde{db})$ and $n(\tilde{db})$ in place of $d_j(db)$ and $n(db)$ respectively.

3.1 Constructing *DF*

The efficacy of our *GlOSS* server depends on how well the DF vector is constructed. A very simple "solution" to constructing the DF vector is to summarize *everything* the system has ever seen. For example, for each Web document from *db* ever brought into the cache, the *GlOSS* server increments the entry $DF(db)[j]$ by one if the word t_j appears in the document. However, this simple strategy incorrectly estimates the DF vector when the same document is accessed repeatedly. Here, let us consider the following examples.

Example 1 (FAQ) A FAQ document on a popular topic may be accessed by a number of users. If the time between successive accesses is long enough such that a copy of the document cannot be found in the cache, then every access to the FAQ will bring the same document into the cache. This causes the *GlOSS* server to increment the document frequencies of the words that appear in the FAQ on *every* access. Essentially, the server is fooled to think that there are multiple documents that contain the same words, while in fact, only one version of the FAQ should be counted.

Example 2 (Scoreboard) A Web page which broadcasts the scoreboard of an on-going NBA game is updated and reloaded many times while the game is being played. Similar to the FAQ example, each access to this scoreboard page causes an erroneous increment to the DF vector. The difference between the FAQ example and the scoreboard example is that while the FAQ rarely changes, some words in the scoreboard page are transient, i.e., there may be significant word changes between successive page updates.

In order for the *GlOSS* server not to over-count the document frequencies, we need to identify whether a Web document has ever been seen before, and what should be added or deducted from the DF vector. For the Scoreboard example, since the inter-retrieval time of the same page is in the order of minutes, the previous outdated page is still in the cache. The *GlOSS* server only has to decrement the document frequencies of the words that appear in the outdated page in the cache, and increment those that appear in the new version. For the FAQ example, the old document has probably been flushed out of the cache already when the new version is requested again. Hence, the *GlOSS* server does not have the information to correctly update the DF vector. To rectify, the *GlOSS* server also maintains two more pieces of information:

A log (LOG) of document records For each document doc_i which is retrieved by the system and identified by its URL, a record r_i of two fields $\langle LS_i, LM_i \rangle$ is put in the LOG. LS_i is the *expected* life span of doc_i while LM_i is the last modified time of doc_i. We will illustrate how these fields are used in the next subsection. To limit the amount of storage taken by the LOG, our prototype only keeps records of Web pages retrieved in the past 400 days. (This window of time can be adjusted.) Loosely speaking, the system *forgets* documents that are more than about a year old. As we will see shortly, the document frequencies due to these old documents are appropriately deducted from the DF vector.

A set of k expiry vectors $V_1, ..., V_k$ For each Web site db, the system keeps a set of expiry vectors $V_i(db)$'s. (In the following discussion, we drop the reference db for simplicity.) When a Web page D is brought into the cache, the document frequencies of the words in D are incremented in the DF vector. The system would then estimate, based on the past history of D, an *expected life span* for D. Essentially, the system assumes that D will be modified on an *expiry day*, and its outdated content should be removed from the DF vector then. When D is subsequently flushed out of the cache, its content is captured by a certain expiry vector of a particular life span. Each expiry vector V_i is associated with an *initial counter* IC_i and an *expiry counter* EC_i. The expiry counter is decremented by one every day. When the expiry counter EC_i ticks down to zero, the corresponding expiry vector V_i is subtracted from the DF vector, signaling that certain Web pages previously accessed by the system are presumed outdated. The system then tries to reset the expiry counter to the value of the initial counter, recycling the expiry vector for capturing other Web pages. Besides IC_i and EC_i, the expiry vector V_i is also associated with a *size counter* $n(V_i)$ which gives the total number of pages V_i summarizes.

To illustrate the use of the LOG and the expiry vectors, we reconsider the previous examples. We only highlight those key steps that show how the system patches the DF vector. Other details are presented in the algorithms shown in the next subsection.

For the FAQ example, if the FAQ document does not change, when it is accessed and brought into the cache for the first time, the DF vector is appropriately updated to reflect the inclusion of the FAQ and a log record is created. The life span of the FAQ is set to a default value. When the FAQ is retrieved the second time, its log record is consulted. Since the *last-modified* time of the newly fetched FAQ is the same as that in the log record, the system knows that the DF vector has already included the content of the FAQ. So, the second retrieval does not induce any update to the DF vector.

For the scoreboard example, we assume that the Proxy cache is big enough to store all the documents the system retrieves in a day. Since the inter-retrieval time of the scoreboard page is very short, when a new version of the same page is retrieved, the previous version can still be found in the cache. The system only has to decrement the document frequencies of the words that appear in the outdated page in the cache, and increment those in the new version.

3.2 Algorithms

In this subsection, we present the algorithms for managing the vectors and the LOG. The algorithms are driven by the following events:

Request When a user requests a document D, the system first checks if the newest version of D is in the cache. If not, D is downloaded from the Web site. If a log record of D is available, that means D has been retrieved before. The system then checks if the newest version of D is the same as the previously one. If not, the DF vector is incremented to include the new D. Otherwise, DF is only updated if the life span was wrongly estimated to have already expired in the previous estimate. In any case, the log record is appropriately updated. If D is retrieved for the first time, a new log record is created and the life span of D is set to a default value. Figure 2 shows the algorithm for handling a user request.

Flush When a document D is to be flushed out of the cache, the system captures D's content in an expiry vector. The expiry date of D is its last modified time plus its life span, and the number of days to expiry equals the expiry date minus the current time. The expiry vector whose counter EC is closest to the number of days to expiry is chosen to capture D. In case the expiry date has already passed, D's content is expired immediately and the DF vector is updated.

End-of-day At the end of a day, the expiry counters ECs' of all the expiry vectors are decremented by one. If an expiry counter EC_i becomes zero, V_i is deducted from the DF vector. The system then *tries* to reset EC_i to its initial value IC_i. However, if there exists another expiry vector V_j whose expiry counter EC_j has the same value as IC_i, there are now two expiry vectors V_i and V_j with the same expiry counter value. We call this a vector clash. Different expiry vectors should have different expiry counters in order to effectively capture

```
Request(D)
{ let db = web site containing D ;
  retreive HTTP header of D from web site db ;
  if (D is not in cache) or
     (header's last modified time <> cache copy's last modified time)
  {  download D from db ; }

  if (LOG[D] does not exist)
  {  create LOG[D] ;
     set LOG[D].LS = default life span ;
     set LOG[D].LM = NULL ; // tentative, to be fixed in (*) below
  }
  else if (LOG[D].LM <> D's last modified time)
  {  set LOG[D].LS = D's last modified time - LOG[D].LM ; }

  // (*) set LM field if not done yet ; update DF vector if necessary
  if (LOG[D].LM <> D's last modified time)
  {  set LOG[D].LM = D's last modified time ;
     if (D is in cache)
     {  for each word tj in D's old version in cache do
        {  DF(db)[j]-- ; }
        n(db)-- ;
     }

     for each word tj in D do
     {  DF(db)[j]++ ; }
     n(db)++ ;
  }
}
```

Fig. 2. Algorithm Request

```
Flush(D)
{ if ( (LOG[D].LM + LOG[D].LS) > current time )
  {  let Vi(db) =
        expiry vector with ECi = LOG[D].LM + LOG[D].LS - current time ;

     for each word tj in D do
     {  Vi(db)[j]++ ; }
     n(Vi(db))++ ;
  }
  else
  {  for each word tj in D do
     {  DF(db)[j]-- ; }
     n(db)-- ;
  }
}
```

Fig. 3. Algorithm Flush

documents that exhibit different life spans. Vector clash is thus undesirable. To ensure the expiry vectors cover a reasonable spectrum of expiry dates, the system dynamically adjusts the vectors' expiry counter values. In particular, when a vector clash happens, the system *swaps* the roles of the clashing vectors. That is, it swaps IC_i with IC_j and tries to reset EC_i to the new IC_i. If V_i then clashes with yet another vector V_k, further swapping is done until there is no vector clash. Figure 4 shows the algorithm for updating the DF and expiry vectors at the end of a day.

```
End-of-day()
{ for each Web site db do
  {  for each expiry vector Vi(db) do { ECi(db)-- ; }
     if (there exists a vector Vi(db) such that ECi(db) = 0)
     {  n(db) = n(db) - n(Vi(db)) ;
        for each non-zero entry k in Vi(db) do
        {  DF(db)[k] = DF(db)[k] - Vi(db)[k] ; }
        while (TRUE) do
        {  ECi(db) = ICi(db) ;
           if (there exists another vector Vj(db) with ECi(db) = ECj(db))
              swap(ICi(db), ICj(db)) ;
           else exit while loop ; } } }
}
```

Fig. 4. Algorithm End-of-day

4 Experiment

To illustrate the effectiveness of our *GlOSS* update algorithm, we implemented a prototype system and conducted an experiment. The goal is to study how well the system keeps the word statistics of a Web site. In this section we briefly discuss the experiment result.

We selected a sports Web site and took a snapshot of the site every day, for 76 consecutive days. Each snapshot has, on the average, 20,000 Web pages. The Web site also has archives of pages which are only accessible via its search interfaces. The snapshots we took did not include those archives. Over the 76 snapshots, there are about 124,000 unique pages, with unique URLs'. Many pages have dynamic contents or exhibit the *transient existence property*. For example, only 416 pages, most of which being index pages, exist throughout the 76 days, and none has its content remain unmodified for more than 21 days.

To model Web page accesses, we first identified the "index pages" of the Web site, e.g., the "soccer" index page, the "tennis" index page, etc. Each index page represents an *area* of interest of the Web site. Each other Web page was then grouped with the index page which shares with it the same URL prefix. Finally,

we identified the pages which exhibit *transient existence* property, i.e., those that only exist for less than two days. We simulated a situation in which 5% of the pages in each snapshot were accessed by the prototype. Since the average snapshot size is 20,000 pages, about 1,000 pages were accessed per snapshot.

We synthesized the page requests according to the following rules:

1. The home page and all the index pages are accessed.
2. Three quarters of the page requests, i.e., about 750, are distributed evenly to each area of interests.
3. One quarter of the page requests, i.e., about 250, are randomly selected from the pages of transient existence.

Rule (1) is used to model the fact that the index pages are usually the most frequently accessed pages. With rule (2), we assume different areas are of similar popularity. With rule (3), we observe that the pages of transient existence usually contain "fresh news" and are more likely be accessed. The pages selected under rule (2) may overlap with those pages selected under rule (3). Thus the total number of pages accessed per snapshot may be slightly less than 1,000.

The prototype system implements all the details as discussed in Section 3. For the experiment, since we have only 76 snapshots of the Web site, we set the default life span of a page to 60 days.

We recall that *GlOSS* uses the fraction of documents in a database for a particular keyword, i.e., $d_i(db)/n(db)$ in Equation 1, to estimate the goodness of the database with respect to a query. We call this fraction the *document density of the i-th word*, and denote it by the symbol $\rho_i(db)$. We will show how accurate the prototype estimates the document densities. Since our *GlOSS* system only "sees" a sample of pages from the Web site, the document densities are just *estimates* based on the sample pages. The document frequency information of the sample pages are summarized in the DF vector while $n(\tilde{db})$ records the total number of pages ever retrieved from the Web site db and not yet expired. The estimated document density of the i-th word $\rho_i^{est}(db)$ is thus $DF(db)[i]/n(\tilde{db})$.

We drove the prototype using the synthesized streams of page requests. We then compared the estimated document densities $\rho_i^{est}(db)$ with the real document densities $\rho_i(db)$ of the words that appear in the last snapshot. In particular, we compute the *relative error*, e_i, defined by:

$$e_i = \left(\rho_i^{est}(db) - \rho_i(db)\right)/\rho_i(db), \qquad \text{for all word } t_i \text{ in the last snapshot.}$$

Figure 4 shows a bar graph which illustrates the average relative error of words whose document densities fall into a specific range. From the figure, we see that the average relative error for words with a non-trivial density ($\rho_i(db) > 5\%$) is within 10%. This shows that our system is able to estimate the document densities and thus the goodness of a Web site from a small set of document samples (5%) fairly accurately. For words with a low document density ($0 \leq \rho_i(db) \leq 0.5\%$), however, the average relative error is not insignificant. Fortunately, these words with such a low document density appear very rarely in the Web site. The error in the estimate of words with low document density is thus unimportant. Hence, it does not undermine the effectiveness of our *GlOSS* system.

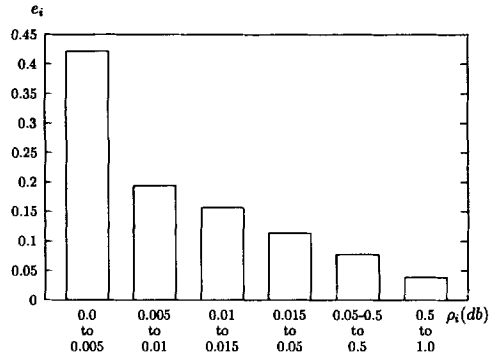


Fig. 5. Relative error versus densities

5 Conclusion

In this paper, we discuss the text-source discovery problem and the *GlOSS* approach. Due to the dynamic nature of the Web, a direct application of *GlOSS* is inadequate. The problem is how one could obtain updated keyword statistics of the Web sites, which are an important piece of information *GlOSS* relies on in making an effective recommendation. We call this problem the *GlOSS update problem.* We discuss an approach to solving the *GlOSS* update problem based on passive sampling of documents from the Web sites. Data structures and algorithms for dynamically tracking the keyword statistics of Web sites are proposed. We implemented the ideas in a prototype *GlOSS* system. We conducted a simple experiment verifying the effectiveness of our approach. The result shows that our system can give a reasonable estimate of document densities. The estimate is particularly accurate for those frequently occuring keywords.

References

1. *The Web Robots FAQ.* URL: http://www.mesquite.com.
2. M.E. Maron D.C. Blair. An evaluation of retrieval effectiveness for a full-text document retrieval system. *Communications of the ACM*, 28(3):290–299, 1985.
3. S. Feldman. Just the answers, please: Choosing a web search service. *The Magazine for Database Professionals*, May 1997.
4. B. Grossan. *Search Engines: What They Are? How They Work?* URL: http://webreference.com/content/search/features.html.
5. V.N. Gudivada. Information retrieval on the world wide web. *IEEE Internet Computing*, 1(5):58–68, 1997.
6. Anthony Tomasic Luis Gravano, Héctor García-Molina. The effectiveness of *GlOSS* for the text-database discovery problem. In *Proceedings of the 1994 ACM SIGMOD.*
7. Anthony Tomasic Luis Gravano, Héctor García-Molina. Generalizing *GlOSS* to vector-space databases and broker hierarchies. In *Proceedings of the 1995 VLDB Conference*, May 1995.

Extraction of Fuzzy Clusters from Weighted Graphs

Seiji Hotta, Kohei Inoue, and Kiichi Urahama

Kyushu Institute of Design, Fukuoka 815-8540, Japan
urahama@kyushu-id.ac.jp

Abstract. A spectral graph method is presented for partitioning of nodes in a graph into fuzzy clusters on the basis of weighted adjacency matrices. Extraction of a fuzzy cluster from a node set is formulated by an eigenvalue problem and clusters are extracted sequentially from major one to minor ones. A clustering scheme is devised at first for undirected graphs and it is next extended to directed graphs and also to undirected bipartite ones. These clustering methods are applied to analysis of a link structure in Web networks and image retrieval queried by keywords or sample images. Extracted structure of clusters is visualized by a multivariate exploration method called the correspondence analysis.

1 Introduction

Data summarization by clustering is a fundamental strategy for exploration of large scale data structures in information retrieval, filtering and data mining[1]. Hypertexts or data matrix in document retrieval can be represented by directed or undirected graphs. Clustering of graphs is useful for exploration of those data structures. Clustering is an NP-hard problem which needs approximate solution methods one of which is spectral graph partitioning[2] in which integer constraints are relaxed to real values and combinatorial tasks are reduced to eigenvalue problems. Spectral partitioning method is recently extended to graphs and matrices[3] and to categorical data[4]. The method[3] yields continuous, i.e. fuzzy clustering instead of discrete, i.e. hard one, however the number of clusters is fixed a priori. The method[4] is an extension of their approach[5] for finding a dense set in directed graphs to a strategy for finding multiple sets by using a nonlinear iteration schemes. We have presented another spectral graph method[6] for partitioning data into fuzzy clusters. Our method applies a linear eigenvalue solution to the adjacency matrix of data successively for extracting fuzzy clusters sequentially. In this paper, we extend it to directed graphs and to undirected bipartite ones and apply them to analysis of a link structures in Web networks and to image retrieval queried by keywords or sample images. Extracted structure of clusters is visualized by a multivariate exploration method called the corresponding analysis. This visualization can be used for image retrieval by browsing of images and keywords.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 442–453, 2000.
© Springer-Verlag Berlin Heidelberg 2000

2 Sequential Extraction of Fuzzy Clusters by Spectral Graph Method

We partition nodes of a graph whose edges and nodes have real weights into a set of fuzzy clusters. Our method[6] for fuzzy clustering of data is first applied to undirected graphs. It is next extended to directed graphs and to undirected bipartite ones. Their practical implication will be clarified in the next section for their applications.

2.1 Fuzzy Clustering of Undirected Graphs

Let the weight of the ith node in an undirected graph of m nodes be v_i, and the weight of the edge between the ith and the jth nodes be w_{ij} (see Fig.1(a)). Undirected graph implies $w_{ij} = w_{ji}$. Our method[6] extracts cohesive subsets successively from this node set. Let x_{1i} be the degree of participation of the ith node in the first, i.e. most densely connected cluster, then the cohesiveness of this cluster is evaluated by $\sum_{i=1}^{m}\sum_{j=1}^{m} v_i x_{1i} w_{ij} v_j x_{1j}$ and the first cluster maximizes it as

$$\begin{aligned} &\max_{x_1} \quad \sum_{i=1}^{m}\sum_{j=1}^{m} v_i x_{1i} w_{ij} v_j x_{1j} \\ &\text{subj.to} \quad \sum_{i=1}^{m} v_i x_{1i}^2 = 1 \end{aligned} \tag{1}$$

This first step of extraction coincides with the method[7] for finding the most cohesive subset in image data. This equation is converted by the transformation of variables $\tilde{x}_{1i} = \sqrt{v_i} x_{1i}$ into a canonical form:

$$\begin{aligned} &\max_{\tilde{x}_1} \quad \sum_{i=1}^{m}\sum_{j=1}^{m} \sqrt{v_i}\tilde{x}_{1i} w_{ij} \sqrt{v_j}\tilde{x}_{1j} \\ &\text{subj.to} \quad \sum_{i=1}^{m} \tilde{x}_{1i}^2 = 1 \end{aligned} \tag{2}$$

The optimal solution $\tilde{x_1} = [\tilde{x}_{11}, ..., \tilde{x}_{1m}]$ of this maximization problem is the principal eigenvector of the weighted adjacency matrix $W_1 = [\sqrt{v_i} w_{ij} \sqrt{v_j}]$.

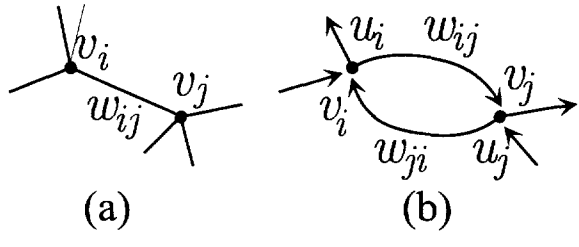


Fig. 1. Undirected graph (a) and directed one (b)

Since all elements of W_1 are nonnegative, every $\tilde{x_{1i}}$ is also nonnegative, and the cohesiveness of the cluster is given by the principal eigenvalue. The solution of eq.(1) is given by $x_{1i} = \tilde{x}_{1i}/\sqrt{v_i}$ and let $i_1 = \arg\max_i x_{1i}$, then $p_{1i} = x_{1i}/x_{1i_1}$ is the membership of the ith node in the first cluster[6].

The second cluster is extracted by the same procedure after the deletion of the first cluster from the original node set. After the deletion the weight of each node becomes $(1 - p_{1i})v_i$, hence the membership x_{2i} of the ith node in the second cluster is given by replacing v_i in x_{1i} by $(1 - p_{1i})v_i$. Thus extracted clusters are deleted successively and then generally the membership p_{ki} in the kth cluster is given by the procedure: we first calculate the principal eigenvector $\tilde{x}_k$ of the matrix $W_k = [\sqrt{v_i v_j} w_{ij} \prod_{l=1}^{k-1} \sqrt{(1-p_{li})(1-p_{lj})}]$, next $x_{ki} = \tilde{x}_{ki}/\sqrt{v_i \prod_{l=1}^{k-1}(1-p_{li})}$ is calculated and $i_k = \arg\max_i x_{ki}$ is found, then the membership of the ith node in the kth cluster is given by $p_{ki} = x_{ki}/x_{ki_k}$. Since every element of the matrix decreases monotonically in this sequential extraction of clusters, its principal eigenvalue, i.e. the cohesiveness also decreases monotonically. The profile of the variation in the cohesiveness suggests us an appropriate number of clusters.

2.2 Fuzzy Clustering of Directed Graphs

Generally $w_{ij} \neq w_{ji}$ in directed graphs. The weight of nodes is differentiated into two, one is the weight u_i as an initial node for the edges outgoing from the ith node and the terminal weight v_i for edges incoming to it(see Fig.1(b)). Accordingly the membership of nodes is divided to the initial membership x_i and the terminal membership y_i. The first cluster with the largest cohesiveness is then formulated as

$$\begin{aligned} \max_{x_1, y_1} \quad & \sum_{i=1}^{m}\sum_{j=1}^{m} u_i x_{1i} w_{ij} v_j y_{1j} \\ \text{subj.to} \quad & \sum_{i=1}^{m} u_i x_{1i}^2 = 1, \quad \sum_{j=1}^{m} v_j y_{ij}^2 = 1 \end{aligned} \tag{3}$$

which is an extension of eq.(1). This first step of cluster extraction coincides with the method[5] for finding the most dense subset in Web networks. Similarly to the previous subsection the variable transformation $\tilde{x}_{1i} = \sqrt{u_i} x_{1i}, \tilde{y}_{1j} = \sqrt{v_j} y_{1j}$ converts eq.(3) into

$$\begin{aligned} \max_{\tilde{x}_1, \tilde{y}_1} \quad & \sum_{i=1}^{m}\sum_{j=1}^{m} \sqrt{u_i}\tilde{x}_{1i} w_{ij} \sqrt{v_j}\tilde{y}_{1j} \\ \text{subj.to} \quad & \sum_{i=1}^{m} \tilde{x}_{1i}^2 = 1, \quad \sum_{j=1}^{m} \tilde{y}_{ij}^2 = 1 \end{aligned} \tag{4}$$

whose solution $\tilde{x}_1 = [\tilde{x}_{11}, ..., \tilde{x}_{1m}], \tilde{y}_1 = [\tilde{y}_{11}, ..., \tilde{y}_{1m}]$ is shown by the Lagrange multiplier method to meet the following two equations:

$$W_1 \tilde{y}_1 = \lambda \tilde{x}_1, \quad W_1^T \tilde{x}_1 = \mu \tilde{y}_1 \tag{5}$$

where $W_1 = [\sqrt{u_i} w_{ij} \sqrt{v_j}]$, and λ and μ are Lagrange multipliers. Elimination of $\tilde{y}$ from these two equations, or conversely the elimination of $\tilde{x}$ leads to

$$W_1 W_1^T \tilde{x}_1 = \lambda\mu \tilde{x}_1, \quad W_1^T W_1 \tilde{y}_1 = \lambda\mu \tilde{y}_1 \tag{6}$$

which state that $\tilde{x}_1$ is the principal eigenvector of $W_1 W_1^T$ and $\tilde{y}_1$ is that of $W_1^T W_1$. In practical computation, one of these two eigenvectors, e.g. $\tilde{y}_1$ is calculated and then the other is given by only matrix multiplication as $\tilde{x}_1 = W_1 \tilde{y}_1$. The normalization of its norm is unnecessary because it will be renormalized at the transformation to memberships. The solution of eq.(3) is $x_{1i} = \tilde{x}_{1i}/\sqrt{u_i}$, $y_{1j} = \tilde{y}_{1j}/\sqrt{v_j}$ from which the memberships are given by $p_{1i} = x_{1i}/\max\{x_{1i}\}$, $q_{1j} = y_{1j}/\max\{y_{1j}\}$ where p_{1i} is the initial membership of the ith node and q_{1j} is the terminal membership of the node j. We call $i_1 = \arg\max_i\{p_{1i}\}$ the representative initial node in the first cluster, which corresponds to the hub in [5], and $j_1 = \arg\max_j\{q_{1j}\}$ is the representative terminal node which is called the authority in [5].

Let us evaluate the cohesiveness of clusters. If we multiply $\tilde{x}_1$ to the left equation of (5), we get $\tilde{x}_1^T W_1 \tilde{y}_1 = \lambda \tilde{x}_1^T \tilde{x}_1 = \lambda$, and similarly the multiplication of $\tilde{y}_1$ to the right equation of (5) leads to $\tilde{y}_1^T W_1^T \tilde{x}_1 = \mu \tilde{y}_1^T \tilde{y}_1 = \mu$. From these two equations we know $\lambda = \mu$, from which together with eq.(6) it is derived that $\lambda(= \mu)$ is the square root of the principal eigenvalue of $W_1^T W_1$, which is equal to that of $W_1 W_1^T$. Thus it is concluded that the cohesiveness $\sum_{i=1}^m \sum_{j=1}^m u_i x_{1i} w_{ij} v_j y_{1j} = \tilde{x}_1^T W_1 \tilde{y}_1$ is given by $\lambda(= \mu)$.

Next at the extraction of the second cluster, the weight of nodes as initial points is reduced to $(1 - p_{1i})u_i$ and their terminal weight becomes $(1 - q_{1j})v_j$. The membership in the second cluster is hence given by the above expression for the first cluster with the replacement of u_i by $(1 - p_{1i})u_i$ and that of v_j by $(1 - q_{1j})v_j$.The third and subsequent clusters are given similarly and generally the kth cluster is calculated by the following procedure: the principal eigenvector $\tilde{y}_k$ of $W_k^T W_k$ where $W_k = [\sqrt{u_i v_j} w_{ij} \prod_{l=1}^{k-1} \sqrt{(1 - p_{li})(1 - q_{lj})}]$ is calculated and from it we get $\tilde{x}_k = W_k \tilde{y}_k$ which is further transformed to $x_{ki} = \tilde{x}_{ki} / \sqrt{u_i \prod_{l=1}^{k-1} (1 - p_{li})}$. The representative initial node i_k is that with maximal x_{ki} and the representative terminal node j_k is with maximal y_{kj}. The membership as initial nodes in the kth cluster is then given by $p_{ki} = x_{ki}/x_{ki_k}$ and that as terminal nodes is $q_{kj} = y_{kj}/y_{kj_k}$. The cohesiveness of the kth cluster is the square root of the principal eigenvalue of $W_k^T W_k$ (or $W_k W_k^T$). The cohesiveness decreases monotonically as k increases.

Finally before proceeding to the next section, we show that directed graphs can be transformed to undirected bipartite graphs equivalent to them. At first each node i is doubled to $\tilde{i}$ and $\hat{i}$ and an edge with the weight w_{ij} is drawn from $\tilde{i}$ to $\hat{i}$. The direction of edges can be dropped and then we get an undirected bipartite graph (see Fig.2) where the weight of node $\tilde{i}$ is u_i and that of $\hat{j}$ is v_j. Clustering of such an undirected bipartite graph is considered in the next section.

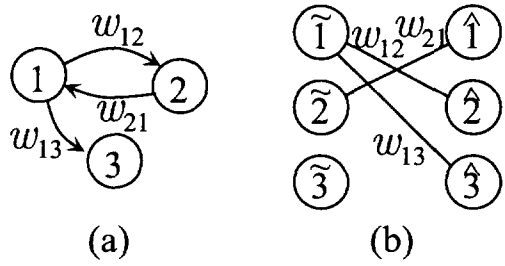


Fig. 2. Undirected bipartite graph (b) equivalent to digraph (a)

2.3 Undirected Bipartite Graphs

Let a graph be composed of two subsets of nodes $i = 1, ..., m$ and $j = 1, ..., n$ between which undirected edges with a weight w_{ij} $(= w_{ji})$ are linked but no edge exists in each subset. There are three cases for the clustering of those undirected bipartite graphs.

The first is clustering of the entire graph as an undirected graph by ignoring the specificity of its bipartite structure. In this case the method in section 2.1 is simply applied to the entire graph.

The second is the clustering of two subsets separately. This clustering cannot be done by applying the method in section 2.1 to each subset independently because there is no direct link in the subset. Let x_1 be the degree of participation of the nodes in one subset in the first cluster and y_1 be that of the nodes in the other subset, then the equation for these variables reads the same as eq.(3). This means the equivalence between directed graphs and undirected bipartite ones as is stated at the end of the previous section. Separate clustering can be recognized by the elimination of x_1 or y_1 from eq.(3), which yields two separate equations

$$\begin{aligned} &\max_{\tilde{x}_1} \quad \tilde{x}_1^T W_1 W_1^T \tilde{x}_1 \\ &\text{subj.to} \quad \| \tilde{x}_1 \| = 1 \end{aligned} \tag{7}$$

$$\begin{aligned} &\max_{\tilde{y}_1} \quad \tilde{y}_1^T W_1^T W_1 \tilde{y}_1 \\ &\text{subj.to} \quad \| \tilde{y}_1 \| = 1 \end{aligned} \tag{8}$$

which reveal that $W_1 W_1^T$ is the adjacency matrix for x_1 and $W_1^T W_1$ is that for y_1. This separation however cannot be recommended from their computational costs. Since the matrix W_1 is generally nonsquare because $m \neq n$ (W_1 is square for directed graphs in section 3.1), the size of $W_1^T W_1$ is different to that of $W_1 W_1^T$, hence the cost is minimized by calculating only the eigenvector of smaller one of these matrices and obtaining the other eigenvector by using one of eq.(5).

Finally the third case is clustering of only one subset of nodes. Contrarily to the second case where clusters are extracted concurrently from both subsets, the other subset is invariant throughout the extraction process in this case. Let the subset $i = 1, ..., m$ be partitioned into clusters. The extraction of the first cluster

is the same as eq.(3). Next at the extraction of the second cluster, only the weight of the subset $i = 1, ..., m$ is reduced to $(1 - p_{1i})u_i$ and v_j $(j = 1, ..., n)$ are fixed invariantly. The extraction of the third and subsequent clusters is similar and generally the kth cluster is calculated by the following procedure: the principal eigenvector $\tilde{x}_k$ of $W_k W_k^T$ where $W_k = [\sqrt{u_i v_j} w_{ij} \prod_{l=1}^{k-1} \sqrt{1 - p_{li}}]$ is calculated (or if $m \geq n$ the principal eigenvector $\tilde{y}_k$ of $W_k^T W_k$is calculated and from it we get $\tilde{x}_k = W_k \tilde{y}_k$) and from it we get $x_{ki} = \tilde{x}_{ki} / \sqrt{u_i \prod_{l=1}^{k-1}(1 - p_{li})}$. The representative node i_k is that with maximal x_{ki} and the membership of the ith node in the kth cluster is given by $p_{ki} = x_{ki}/x_{ki_k}$. The cohesiveness of the kth cluster is the square root of the principal eigenvalue of $W_k^T W_k$ (or $W_k W_k^T$) and decreases monotonically as k increases. Note that the membership in extracted clusters is obtained for the nodes $j = 1, ..., n$ as a byproduct of this clustering of the subset $i = 1, ..., m$. The representative node j_k of the kth cluster is the one with maximal y_{kj} and the membership is given by $q_{kj} = y_{kj}/y_{kj_k}$ for the jth node in the kth cluster.

3 Applications

We next proceed from the basic theory of the clustering algorithms described above to some examples of their practical applications. The order of subsections below corresponds to that of subsections in the above section.

3.1 Clustering of Metric Data

When data are given by a set of points in a metric, e.g. Euclidean spaces, the data are represented by an undirected graph in which each node denotes a datum and edges have the weight which is the inverse of the distance between data. Their clustering is obtained by the procedure described in section 2.1. This class of data is very popular and their clustering is rather elementary, hence its examination is skipped here.

3.2 Clustering of Web Pages

Next examples are hypertexts and Web pages which are represented by a directed graph where each text or page is denoted by a node and links are represented by directed edges between nodes. A set of journal articles are also represented by a directed graph where the links denote citational relations. These examples are analyzed by the method in section 2.2. We report here the clustering of Web pages. We used a set of 127 Web pages which are retrieved by a search engine "metacrawler" with the keyword "information retrieval" together with secondary searches. A directed graph was constructed from the links in those pages. All the weights were set to one for both nodes and edges. Figure 3 shows the cohesiveness of extracted clusters. Its decreasing rate deteriorates after the 4th extraction, this shows that clusters from the 4th one are sparse and the number

of salient clusters is three. Their representative initial pages (whose URL and title) are

1. http://cuiwww.unige.ch/ viper/other-systems.html
   Links to other image database systems
2. http://n106.is.tokushima-u.ac.jp/member/kita/NLP/IR.html
   Information Retrieval Links
3. http://www.umiacs.umd.edu/research/CLIP/filter2.html
   Information Filtering Resources

and their representative terminal pages are

1. http://wwwqbic.almaden.ibm.com/
   QBIC - IBM's Query By Image Content
2. http://ciir.cs.umass.edu/
   The Center for Intelligent Information Retrieval
3. http://www.lucifer.com/ sasha/articles/ACF.html
   Automated Collaborative Filtering and Semantic Transports

In these lists the number is the extracted order which is upper for the cluster including more pages. The representative initial page called the hub in [5] has links to many pages in the cluster, hence it serves to introduction to the topics of the cluster. The representative terminal page called the authority in [5] is linked from many pages in the cluster, hence its content is important for the topics. The topics of above three clusters are 1. image retrieval, 2. information retrieval, and 3. information filtering. Though this example contains very small number of pages of restricted topics, clustering of arbitrarily gathered pages of large numbers could find groups emergently build up in the internet such as a cyber-community.

The present cluster analysis of Web networks can be used for suggestion of new links for each page. Let us consider an intuitive example where the initial membership p_{ki} of a page in a cluster has the value one and p_{li} $(l \neq k)$ of that page in other clusters are zero. This means that some links exist from that page

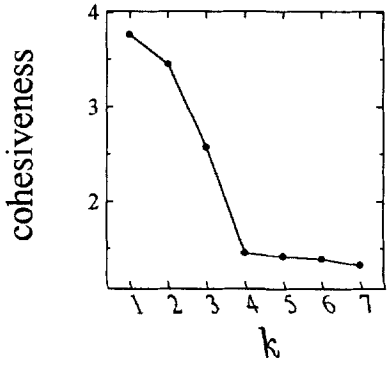


Fig. 3. Variation in cohesiveness of clusters

to some pages included in that cluster and there is no link from that page to other clusters. Then a new link from that page to the representative terminal page of that cluster is highly recommended if it does not exist yet. By extending this intuitive example with crisp membership values to general cases with fuzzy memberships, we get the following recommendation rule. Let $p_i = [p_{1i}, ..., p_{Ni}]^T$ where N is the number of clusters be the vector of the membership of the ith page as initial nodes in each cluster and $q_j = [q_{1j}, ..., q_{Nj}]^T$ be that of the jth page as terminal nodes. Then the degree of the desirability of the link from the ith page to the jth page can be estimated by the similarity between these two vectors. For instance, the similarity can be evaluated by the cosine of the angle between these two vectors: $r_{ij} = p_i^T q_j / \parallel p_i \parallel \parallel q_j \parallel$ where $p^T q_i$ is the inner product $\sum_{k=1}^{N} p_k q_{ki}$ and $\parallel p \parallel$ is the Euclidean norm $\sqrt{p^T p}$. Thus if there is no link from the ith page to the jth one and r_{ij} is large, then this link is recommended to be added newly.

3.3 Image Retrieval by Keywords

Next is the third example of data retrieval based on the clustering to which the scheme in section 2.3 can be applied. Data in the document retrieval by keywords can be represented by an undirected bipartite graph in which the set of documents is one subset of nodes and keywords are denoted by the other subset of nodes and the weight of links denotes the frequency of keywords appearing in each document. For instance a data matrix in table 1 is represented by the bipartite graph shown in Fig.4 in which all the weights of nodes and those of edges are one. By clustering the documents using the method in section 2.3, we can

Table 1. A data matrix for texts and keywords

	kw1	kw2	kw3	kw4
text1	1	0	1	0
text2	1	1	0	1
text3	0	1	0	1

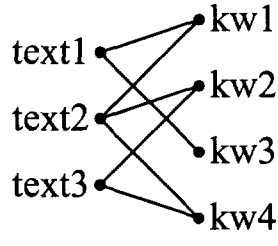


Fig. 4. Graph representation for table 1

grasp the cluster structure of documents and at the same time the membership of each keyword in each cluster is also obtained as a byproduct. Documents and keywords are related through these memberships and we can retrieve documents queried by keywords by referring these relations. Fuzziness of the clustering enables us detailed ranking which is impossible by hard clustering. Objects in a database are not restricted to documents. We examine image databases in this section.

Let us first consider the retrieval by query. Another strategy of retrieval by browsing will be shown in the next section. Let us begin with a simple case with a query being one keyword whose membership in each cluster is p_k $(k = 1, ..., N)$ where N is the number of clusters. Let q_{ki} be the membership of the ith image in the kth cluster. Then the query keyword is represented by the vector $p = [p_1, ..., p_N]^T$ and each image is represented by the vector $q_i = [q_{1i}, ..., q_{Ni}]^T$. Intuitive is the hard clustering case where images included in a cluster to which the query belongs are retrieved. This rule is extended to fuzzy clustering cases similarly to the above example for Web link recommendation. We evaluate the similarity between the query and each image by the cosine of the angle between their vectors: $s_i = p^T q_i / \parallel p \parallel \parallel q_i \parallel$ where $p^T q_i$ is the inner product $\sum_{k=1}^{N} p_k q_{ki}$ and $\parallel p \parallel$ is the Euclidean norm $\sqrt{p^T p}$. This similarity is the correlation coefficient between the query and each image. Images with large similarity to the query are relevant to that query, hence the rank of the relevance of images to the query is given by s_i. Thus images are retrieved by the order of the value of s_i. This retrieval scheme can be extended to a query given by multiple keywords. When L keywords are inputted, the cosine $s_{li} = p_l^T q_i / \parallel p_l \parallel \parallel q_i \parallel$ between the membership vector $p_l = [p_{1l}, ..., p_{Nl}]^T$ of the lth keyword and $q_i = [q_{1i}, ..., q_{Ni}]^T$ of each image is calculated for each keyword $l = 1, ..., L$. These elementary scores are combined by the product $s_i = \prod_{l=1}^{L} s_{li}$ if the combination rule is "AND", or by the sum $s_i = \sum_{l=1}^{L} s_{li}$ if the rule is "OR". Images are retrieved by the order of the value of this combined score s_i.

When the query is given by an image instead of keywords and we search in a database for images similar to the query image, the retrieval process is similar, i.e. the similarity between the query image and each image in the database is evaluated by the cosine of the angle of vectors of those memberships in each cluster.

We have experimented the retrieval of images queried by a sample image. Example database is composed of 160 photographs attached with 46 keywords. This dataset of images was clustered by the scheme in section 2.3. The variation in the cohesiveness of clusters is illustrated in Fig.5 from which the number of clusters is determined to five. Ten images are retrieved for each image in the database as the query. To evaluate the performance of the present retrieval method, we have also experimented the retrieval by using the latent semantic indexing (LSI) method[8]. The rank of the data matrix of the size 160×46 was reduced to five by the singular value decomposition. Ten images are retrieved for each image. Figure 6 shows the number of the same images retrieved both by the present method and by the LSI for each query image. The axis of abscissas

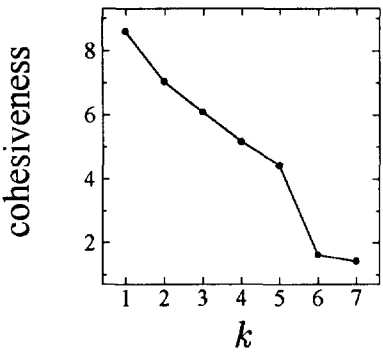


Fig. 5. Variation in cohesiveness of clusters

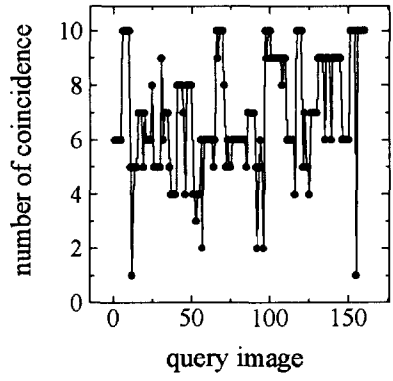


Fig. 6. Number of images appearing both in the first 10 by the present method and in those by the LSI method

in Fig.6 denotes each query image and its axis of ordinates is the number of images coincident in both ten retrieved images. The average number of coincidence is seven which reveals that the present method outputs images similar to those by the LSI method. The entire retrieval time, which is clustering plus 160 retrievals in the present method and singular value decomposition plus 160 retrievals in the LSI, is 0.22 seconds in the present method and 0.33 seconds in the LSI method. Though the present method contains five times of eigenvalue decomposition which is only one time in the LSI method, the present method is faster than the LSI. This is attributed to the fact that the eigendecomposition is applied to a 46×46 matrix in the present method, on the other hand the matrix size is 160×46 in the LSI.

4 Graph Drawing Based on Fuzzy Clustering

Since the membership value in hard clustering takes only one or zero, detailed topology in each cluster is lost hence the ordering of data in each cluster has no meaning. Contrarily the memberships take continuous values in the fuzzy clustering, therefore the locational relation between data displayed on a screen reflects data topology. Two data with similar membership patterns should be placed close together on a screen. Here we calculate approximately such a topology preserving map by using an exploratory multivariate data analysis method called the correspondence analysis[9].

The correspondence analysis is a method for displaying data by arranging them preserving their similarity relations given by their cooccurrence matrix. Let there be m samples and n items and the cooccurrence degree of the ith sample and the jth item be d_{ij}. The procedure of the correspondence analysis for arranging samples on a two-dimensional space based on this cooccurrence matrix $D = [d_{ij}]$ is summarized as follows: If $m \leq n$ then we calculate the secondary

principal eigenvector u_2 and the third one u_3 of the matrix $F^{-\frac{1}{2}}DG^{-1}D^TF^{-\frac{1}{2}}$ where $F = \mathrm{diag}(f_i); f_i = \sum_{j=1}^{n} d_{ij}$ and $G = \mathrm{diag}(g_j); g_j = \sum_{i=1}^{m} d_{ij}$ and next calculate $x = F^{-\frac{1}{2}}u_2$ and $y = F^{-\frac{1}{2}}u_3$. Conversely if $m > n$ then we calculate the second and the third eigenvectors v_2, v_3 and their eigenvectors λ_2, λ_3 of the matrix $G^{-\frac{1}{2}}D^TF^{-1}DG^{-\frac{1}{2}}$, and from them we next calculate $x = F^{-1}DG^{-\frac{1}{2}}v_2/\sqrt{\lambda_2}$ and $y = F^{-1}DG^{-\frac{1}{2}}v_3/\sqrt{\lambda_3}$. These x and y give the locations of data, i.e. (x_i, y_i) is the two-dimensional coordinate of the ith sample.

We exploit this correspondence analysis here for the visualization of the structure of extracted fuzzy clusters. We regard the nodes in a graph as samples and their cluster as an item, then the degree of cooccurrence d_{ij} corresponds to the degree of the ith node appearing in the jth cluster, i.e. the membership of the ith node in the jth cluster. Hence we construct the cooccurrence matrix D by equating d_{ij} to p_{ji} which is the membership of the ith node in the jth cluster and apply the correspondence analysis to it, then two nodes with similar membership values are located close together and we can grasp the cluster structure visually. For example five clusters extracted from 160 images and 46 keywords cited in the previous section were displayed by this method. In this example the samples are images and keywords, and items are clusters, hence $m \gg n$ which is the latter case in the above procedures. Resulted arrangement of images and keywords is illustrated in Fig.7(a) where images are denoted by white squares and the black dots denote keywords, and representative images and keywords are displayed large. Links in Fig.7(a) denote the correspondence between images

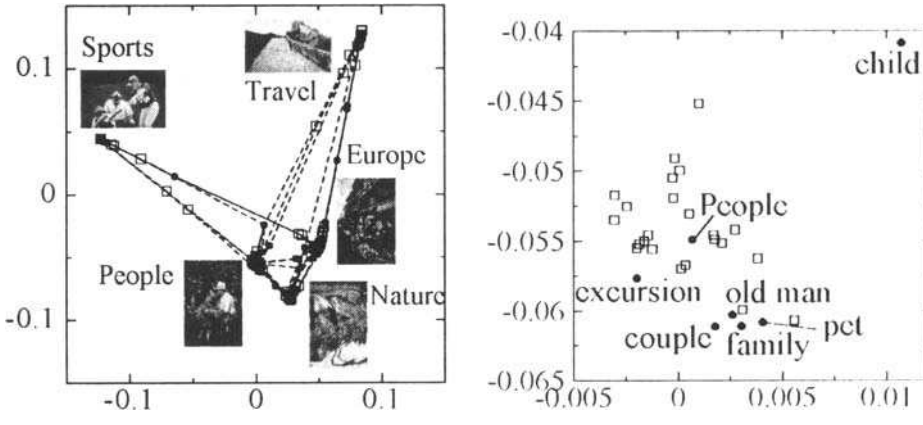


Fig. 7. Drawing of cluster structure for images and keywords

and keywords. The "People" cluster nearly central in Fig.7(a) is magnified in Fig.7(b) where links are omitted for simplification. These simultaneous displays of images and keywords on the same screen serve to grasp these relations and are used for the browsing retrieval of images with the aid of the search based on keywords. On the other hand however such simultaneous display is rather

complicated to see. Search speed will be raised in some cases if only keywords are displayed first and we select a place in the screen and then the images in the selected area are displayed on the screen. Although the present display method is different to the graph drawing schemes[10] whose object is the visibility of the link structure of graphs because our main concern here is the presentation of the cluster structure, the present scheme can be said one of the graph drawing method with the emphasis on the cluster structure.

5 Conclusion

A spectral graph method has been presented for partitioning the node set of a graph into fuzzy clusters on the basis of its adjacency matrix. The method has been applied to the extraction of link structures in Web networks, retrieval of images queried by keywords or sample images and graph drawing based on the clustering. The present method is based on only the link relations between nodes in contrast to conventional clustering method based on only the features of nodes. Combination of these two approaches to clustering is under study to improve their performance.

References

1. Florescu D., Levy A., Mendelzon A.: Database techniques for the World-Wide Web: A survey. SIGMOD Record **27** (1998) 59–74
2. Donath W. E., Hoffman A. J.: Lower bounds for the partitioning of graphs. IBM J. Res. Develop. **17** (1973) 420-425
3. Drineas P., Frieze A., Kannan R., Vempala S., Vinay V.: Clustering in large graphs and matrices. Proc. SODA'99 (1999) 291–299
4. Gibson D., Kleinberg J., Raghavan P.: Clustering categorical data: An approach based on dynamical systems. Proc. 24th VLDB (1998) 311–322
5. Kleinberg J. M.: Authoritative sources in a hyperlinked environment. Proc. SODA'98 (1998) 668–677
6. Inoue K., Urahama K.: Sequential fuzzy cluster extraction by a graph spectral method. Patt. Recog. Lett. **20** (1999) 699–705
7. Sarkar S., Boyer K. L.: Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. Comput. Vision Imag. Und. **71** (1998) 110–136
8. Deerwester S., Dumais S., Furnas G., Landauer T., Harshamn R.: Indexing by latent semantic analysis. J. Amer. Soc. Infor. Sci. **41** (1990) 391–407
9. Jambu M.: Exploratory and multivariate data analysis. Academic Press (1991)
10. Tamassia R., Tollis I. G.: Graph drawing: Algorithms for the visualization of graphs. Prentice Hall (1998)

Text Summarization by Sentence Segment Extraction Using Machine Learning Algorithms

Wesley T. Chuang[1,2] and Jihoon Yang[2]

[1] Dept. of Computer Science, UCLA, Los Angeles, CA, 90095, USA
yelsew@cs.ucla.edu
[2] HRL Laboratories, LLC, 3011 Malibu Canyon Rd, Malibu, CA 90265, USA
{yelsew|yang}@wins.hrl.com

Abstract. We present an approach to the design of an automatic text summarizer that generates a summary by extracting sentence segments. First, sentences are broken into segments by special cue markers. Each segment is represented by a set of predefined features (e.g. location of the segment, number of title words in the segment). Then supervised learning algorithms are used to train the summarizer to extract *important* sentence segments, based on the feature vector. Results of experiments indicate that the performance of the proposed approach compares quite favorably with other approaches (including MS Word summarizer).

1 Introduction

With lots of information pouring in everyday, document summaries are becoming essential. Instead of having to go through the entire text, people can understand the text fast and easily by a concise summary. In order to obtain a good summary, however, we are faced with several challenges. The first challenge is the extent to which we "understand" the text for writing a summary. In our approach we do not claim to generate a summary by *abstract* (after understanding the whole text), but attempt to *extract* some segments out of the text. Second, a summary may be subjective depending on whether people think that certain features or characteristics should be used to generate a summary. Features all have a degree of *subjectivity* and *generality* when they are to be used to form a summary. Ideally, we are looking for those features that are independent of the types of text and users to suggest the significance of parts of a document.

Against this background, we propose an approach to automatic text summarization by *sentence segment extraction* using machine learning algorithms. We perform a "shallow parsing" [5] by looking at special markers in order to determine the sentence segments. Special markers and their by-product, rhetorical relations, discriminate one portion of text from another. We define a set of features for each sentence segment. Then we convert these features into a vector representation and apply machine learning algorithms in order to derive the rules or conditions by which we generate a summary. As we shall find out, machine learning will report to us whether one feature is useful at all in looking for summary material based on achieving a good balance between subjectivity and generality of summarization.

T. Terano, H.Liu, and A.L.P. Chen (Eds.): PAKDD 2000, LNAI 1805, pp. 454–457, 2000.
© Springer-Verlag Berlin Heidelberg 2000

2 Design of the Text Summarizer

2.1 Sentence Segmentation

Our segmentation method is basically the same as Marcu's [5]. A sentence is segmented by a *cue phrase*. The basic idea behind this is to separate units (i.e. sentence segments) that possibly convey independent meanings and use the units in summarization. (See [5] for detailed description on the cue phrases and the segmentation algorithm.)

2.2 Feature Representation

The sentence segments need to be represented by a set of features. There are two kinds of features we consider: *structured* and *non-structured*. The former is related to the structure of the text (e.g. rhetorical relations), while the latter is not (e.g. title words).

Mann and Thompson noted in their *Rhetorical Structure Theory* that a sentence can be decomposed into segments, usually clauses [4]. The main segment is called a *nucleus*, and its subordinate segment is called a *satellite*, relating to the main segment with some rhetorical relation. There are many rhetorical relations signaled by different cue phrases (e.g. *because, but, if, however*). Generally, when a rhetorical relation occurs, the nucleus is considered as a more important segment - and has more chances to be in the summary - than its satellite counterpart.

Using Marcu's discourse-marker-based hypothesizing algorithm [5], we discover rhetorical relations on segments' base level. In other words, we obtain the rhetorical relations of a segment to another segment in nearby region instead of taking all the combinations of segments recursively to generate the whole RST tree, which is computationally demanding.

We collect a total of 23 features and generate the feature vector $\boldsymbol{F}$ which can be divided into three groups:

- Group I: 1. paragraph number, 2. offset in the paragraph, 3. number of bonus words, 4. number of title words, 5. term frequency,
- Group II: 6. antithesis, 7. cause, 8. circumstances, 9. concession, 10. condition, 11. contrast, 12. detail, 13. elaboration, 14. example, 15. justification, 16. means, 17. otherwise, 18. purpose, 19. reason, 20. summary relation,
- Group III: 21. weight of nucleus, 22. weight of satellite, 23. max level.

Features 1-5 in Group I are non-structural attributes of the text. They are counters associated with a feature like the location of the segment as defined in [3,1]. Features 6-20 are distinct rhetorical relations. When a segment is hypothesized with a relation, such a relation (feature) F_i (initially zero) will change its value by the following equation:

$$F_i = \begin{cases} F_i + 1.0/x & \text{if nucleus} \\ F_i - 1.0/x & \text{if satellite} \end{cases}$$

where x is the number of the asymmetric, exclusive-or relations being hypothesized with the segment. Features 21-23 in the last group are collective descriptions of the rhetorical relations. For example, `Weight of nucleus` sums up all the occurrences one segment acting as a nucleus, regardless of which relation it possesses. `Max level` describes how many times, recursively, a segment can be a satellite of another satellite.

2.3 Summarizer Training

Here the goal is to select a few segments as a summary that can represent the original text. With the feature vectors generated in previous steps, we can easily apply machine learning algorithms to train a summarizer (i.e. supervised learning). We are interested in seeing whether programs can quickly learn from our model summary and categorize which segments should be in summary and which should not - and learn it from all 23 aforementioned features that are deemed representative. We consider the following three learning algorithms in our experiments.

Decision Trees We adopted the C4.5, a decision tree learning algorithm designed by Quinlan [6], to train the summarizer. It generates a decision tree making use of the features which give the maximal information gain. C4.5 has been known to be a very fast and efficient algorithm with good generalization capability. (See [6] for detailed descriptions on the algorithm.)

Naive Bayesian Classifier We apply the naive Bayesian classifier as used in [2]: $P(c \in C \mid F_1, F_1, \ldots F_k) = \frac{\Pi_{j=1}^{k} P(F_j|c \in C) P(c \in C)}{\Pi_{j=1}^{k} P(F_j)}$ where C is the set of target classes (i.e. in the summary or not in the summary) and F is the set of features. In our experiment, since the value of the most features are *real* numbers, we assume a *normal distribution* for every feature, and use *normal distribution density function* to calculate the probability $P(F_j)$.

DistAl DistAl [7] is a simple and relatively fast but efficient constructive neural network learning algorithm for pattern classification. (See [7] for detailed description on the algorithm and performance evaluations.)

3 Experimental Results

Among the variety of data on the Internet, we chose nine U.S. patents for our experiments. Note that we used only the sections of "*background of invention*" and "*summary of invention*" instead of considering the entire patent. For each patent data, we manually generated a model summary to be used in training and evaluation of the summarizer. Table 1 displays the number of sentences, number of segments, and the number of segments in the model summary.

Table 1. Dataset size and performance. *sen*, *seg*, and *sum* are the number of sentences and segments in the patent data, and the number of segments in the model summary. *a*, *p*, and *r* represents the accuracy, precision, and recall, respectively (in percentage).

	Data Size			MS Word			C4.5			Bayesian			DistAl		
ID	*sen*	*seg*	*sum*	*a*	*p*	*r*	*a*	*p*	*r*	*a*	*p*	*r*	*a*	*p*	*r*
1	58	75	25	62.7	46.4	50.0	70.7	32.0	61.5	72.0	60.0	48.0	72.0	44.4	48.0
2	29	33	14	48.5	40.0	42.8	60.6	21.4	60.0	60.6	52.9	64.3	57.6	40.0	28.6
3	36	48	16	33.3	23.5	25.0	64.6	37.5	46.1	60.4	41.2	43.8	72.9	45.5	31 .3
4	45	77	20	64.9	33.3	35.0	88.3	60.0	92.3	75.3	52.6	50.0	76.6	47.6	50.0
5	16	19	5	57.4	28.6	40.0	68.4	60.0	42.9	63.2	37.5	60.0	84.2	60.0	60.0
6	95	139	17	76.3	5.6	5.9	82.7	29.4	29.4	84.2	40.0	58.8	87.8	28.9	64.7
7	76	98	25	61.2	25.9	28.0	70.4	32.0	40.0	79.6	60.9	56.0	74.5	33.3	40.0
8	23	29	6	75.9	40.0	33.3	75.8	71.4	71.4	51.7	50.0	71.4	100	100	100
9	30	39	11	66.7	40.0	36.4	71.8	9.1	50.0	76.9	100	18.2	77.0	54.5	54.5
Average				60.8	31.6	32.9	72.6	39.0	54.8	69.3	55.0	52.3	78.1	50. 5	53.0
Standard dev.				13.4	12.1	12.6	8.6	20.0	18.9	10.8	18.8	15.3	11.8	20.9	21.4

The performance of the summarizer is evaluated by a 9-fold cross-validation using the patent data. We evaluate the results of summarization by the *precision*, *recall* and classification *accuracy* (overall percentage of correct classification regardless of the class labels) for the three learning algorithms, and compare them with the summarizer in Microsoft Word. Table 1 displays the performance of all the methods considered. As we can see from Table 1, all the three approaches using machine learning outperformed Microsoft Word summarizer significantly. We do not know exactly the underlying mechanism that Microsoft Word uses to summarize a document. It appears that many co-occurred words are simply selected as a summary. The summary generated by our approach is more coherent than those incoherent fragments of words generated by Microsoft Word summarizer.

References

1. H. P. Edmundson. New Methods in Automatic Extracting. In *Advances In Automatic Text Summarization*, pages 23-42, 1999.
2. J. Kupiec, J. Pedersen, F. Chen. A Trainable Document Summarizer. In *Advances In Automatic Text Summarization*, pages 55-60, 1999.
3. H. P. Luhn. The Automatic Creation of Literature Abstracts. In *Advances In Automatic Text Summarization*, pages 15-21, 1999.
4. W. Mann, S. Thompson. Rhetorical structure theory: Toward a functional theory of text. In *Text* 8(3): pages 243-281, 1988.
5. D. Marcu. *The rhetorical parsing, summarization, and generation of natural language texts.* Ph.D. Dissertation, Department of Computer Science, University of Toronto. 1997.
6. J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
7. J. Yang, R. Parekh, V. Honavar. DistAl: An Inter-pattern Distance-based Constructive Learning Algorithm. In *Intelligent Data Analysis* 3: pages 55-73, 1999.

Author Index

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 1640: W. Tepfenhart, W. Cyre (Eds.), Conceptual Structures: Standards and Practices. Proceedings, 1999. XII, 515 pages. 1999.

Vol. 1647: F.J. Garijo, M. Boman (Eds.), Multi-Agent System Engineering. Proceedings, 1999. X, 233 pages. 1999.

Vol. 1650: K.-D. Althoff, R. Bergmann, L.K. Branting (Eds.), Case-Based Reasoning Research and Development. Proceedings, 1999. XII, 598 pages. 1999.

Vol. 1652: M. Klusch, O.M. Shehory, G. Weiss (Eds.), Cooperative Information Agents III. Proceedings, 1999. XI, 404 pages. 1999.

Vol. 1669: X.-S. Gao, D. Wang, L. Yang (Eds.), Automated Deduction in Geometry. Proceedings, 1998. VII, 287 pages. 1999.

Vol. 1674: D. Floreano, J.-D. Nicoud, F. Mondada (Eds.), Advances in Artificial Life. Proceedings, 1999. XVI, 737 pages. 1999.

Vol. 1688: P. Bouquet, L. Serafini, P. Brézillon, M. Benerecetti, F. Castellani (Eds.), Modeling and Using Context. Proceedings, 1999. XII, 528 pages. 1999.

Vol. 1692: V. Matoušek, P. Mautner, J. Ocelíková, P. Sojka (Eds.), Text, Speech, and Dialogue. Proceedings, 1999. XI, 396 pages. 1999.

Vol. 1695: P. Barahona, J.J. Alferes (Eds.), Progress in Artificial Intelligence. Proceedings, 1999. XI, 385 pages. 1999.

Vol. 1699: S. Albayrak (Ed.), Intelligent Agents for Telecommunication Applications. Proceedings, 1999. IX, 191 pages. 1999.

Vol. 1701: W. Burgard, T. Christaller, A.B. Cremers (Eds.), KI-99: Advances in Artificial Intelligence. Proceedings, 1999. XI, 311 pages. 1999.

Vol. 1704: Jan M. Żytkow, J. Rauch (Eds.), Principles of Data Mining and Knowledge Discovery. Proceedings, 1999. XIV, 593 pages. 1999.

Vol. 1705: H. Ganzinger, D. McAllester, A. Voronkov (Eds.), Logic for Programming and Automated Reasoning. Proceedings, 1999. XII, 397 pages. 1999.

Vol. 1711: N. Zhong, A. Skowron, S. Ohsuga (Eds.), New Directions in Rough Sets, Data Mining, and Granular-Soft Computing. Proceedings, 1999. XIV, 558 pages. 1999.

Vol. 1712: H. Boley, A Tight, Practical Integration of Relations and Functions. XI, 169 pages. 1999.

Vol. 1714: M.T. Pazienza (Eds.), Information Extraction. IX, 165 pages. 1999.

Vol. 1715: P. Perner, M. Petrou (Eds.), Machine Learning and Data Mining in Pattern Recognition. Proceedings, 1999. VIII, 217 pages. 1999.

Vol. 1720: O. Watanabe, T. Yokomori (Eds.), Algorithmic Learning Theory. Proceedings, 1999. XI, 365 pages. 1999.

Vol. 1721: S. Arikawa, K. Furukawa (Eds.), Discovery Science. Proceedings, 1999. XI, 374 pages. 1999.

Vol. 1730: M. Gelfond, N. Leone, G. Pfeifer (Eds.), Logic Programming and Nonmonotonic Reasoning. Proceedings, 1999. XI, 391 pages. 1999.

Vol. 1733: H. Nakashima, C. Zhang (Eds.), Approaches to Intelligent Agents. Proceedings, 1999. XII, 241 pages. 1999.

Vol. 1735: J.W. Amtrup, Incremental Speech Translation. XV, 200 pages. 1999.

Vol. 1739: A. Braffort, R. Gherbi, S. Gibet, J. Richardson, D. Teil (Eds.), Gesture-Based Communication in Human-Computer Interaction. Proceedings, 1999. XI, 333 pages. 1999.

Vol. 1744: S. Staab, Grading Knowledge: Extracting Degree Information from Texts. X, 187 pages. 1999.

Vol. 1747: N. Foo (Ed.), Adavanced Topics in Artificial Intelligence. Proceedings, 1999. XV, 500 pages. 1999.

Vol. 1757: N.R. Jennings, Y. Lespérance (Eds.), Intelligent Agents VI. Proceedings, 1999. XII, 380 pages. 2000.

Vol. 1759: M.J. Zaki, C.-T. Ho (Eds.), Large-Scale Parallel Data Mining. VIII, 261 pages. 2000.

Vol. 1760: J.-J. Ch. Meyer, P.-Y. Schobbens (Eds.), Formal Models of Agents. Poceedings. VIII, 253 pages. 1999.

Vol. 1761: R. Caferra, G. Salzer (Eds.), Automated Deduction in Classical and Non-Classical Logics. Proceedings. VIII, 299 pages. 2000.

Vol. 1771: P. Lambrix, Part-Whole Reasoning in an Object-Centered Framework. XII, 195 pages. 2000.

Vol. 1772: M. Beetz, Concurrent Reactive Plans. XVI, 213 pages. 2000.

Vol. 1778: S. Wermter, R. Sun (Eds.), Hybrid Neural Systems. IX, 403 pages. 2000.

Vol. 1792: E. Lamma, P. Mello (Eds.), AI*IA 99: Advances in Artificial Intelligence. Proceedings, 1999. XI, 392 pages. 2000.

Vol. 1793: O. Cairo, L.E. Sucar, F.J. Cantu (Eds.), MICAI 2000: Advances in Artificial Intelligence. Proceedings, 2000. XIV, 750 pages. 2000.

Vol. 1794: H. Kirchner, C. Ringeissen (Eds.), Frontiers of Combining Systems. Proceedings, 2000. X, 291 pages. 2000.

Vol. 1805: T. Terano, H. Liu, A.L.P. Chen (Eds.), Knowledge Discovery and Data Mining. Proceedings, 2000. XIV, 460 pages. 2000.

Lecture Notes in Computer Science

Vol. 1758: H. Heys, C. Adams (Eds.), Selected Areas in Cryptography. Proceedings, 1999. VIII, 243 pages. 2000.

Vol. 1759: M.J. Zaki, C.-T. Ho (Eds.), Large-Scale Parallel Data Mining. VIII, 261 pages. 2000. (Subseries LNAI).

Vol. 1760: J.-J. Ch. Meyer, P.-Y. Schobbens (Eds.), Formal Models of Agents. Poceedings. VIII, 253 pages. 1999. (Subseries LNAI).

Vol. 1761: R. Caferra, G. Salzer (Eds.), Automated Deduction in Classical and Non-Classical Logics. Proceedings. VIII, 299 pages. 2000. (Subseries LNAI).

Vol. 1762: K.-D. Schewe, B. Thalheim (Eds.), Foundations of Information and Knowledge Systems. Proceedings, 2000. X, 305 pages. 2000.

Vol. 1763: J. Akiyama, M. Kano, M. Urabe (Eds.), Discrete and Computational Geometry. Proceedings, 1998. VIII, 333 pages. 2000.

Vol. 1764: H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg (Eds.), Theory and Application to Graph Transformations. Proceedings, 1998. IX, 490 pages. 2000.

Vol. 1765: T. Ishida, K. Isbister (Eds.), Digital Cities. IX, 444 pages. 2000.

Vol. 1767: G. Bongiovanni, G. Gambosi, R. Petreschi (Eds.), Algorithms and Complexity. Proceedings, 2000. VIII, 317 pages. 2000.

Vol. 1768: A. Pfitzmann (Ed.), Information Hiding. Proceedings, 1999. IX, 492 pages. 2000.

Vol. 1769: G. Haring, C. Lindemann, M. Reiser (Eds.), Performance Evaluation: Origins and Directions. X, 529 pages. 2000.

Vol. 1770: H. Reichel, S. Tison (Eds.), STACS 2000. Proceedings, 2000. XIV, 662 pages. 2000.

Vol. 1771: P. Lambrix, Part-Whole Reasoning in an Object-Centered Framework. XII, 195 pages. 2000. (Subseries LNAI).

Vol. 1772: M. Beetz, Concurrent Reactive Plans. XVI, 213 pages. 2000. (Subseries LNAI).

Vol. 1773: G. Saake, K. Schwarz, C. Türker (Eds.), Transactions and Database Dynamics. Proceedings, 1999. VIII, 247 pages. 2000.

Vol. 1774: J. Delgado, G.D. Stamoulis, A. Mullery, D. Prevedourou, K. Start (Eds.), Telecommunications and IT Convergence Towards Service E-volution. Proceedings, 2000. XIII, 350 pages. 2000.

Vol. 1776: G.H. Gonnet, D. Panario, A. Viola (Eds.), LATIN 2000: Theoretical Informatics. Proceedings, 2000. XIV, 484 pages. 2000.

Vol. 1777: C. Zaniolo, P.C. Lockemann, M.H. Scholl, T. Grust (Eds.), Advances in Database Technology – EDBT 2000. Proceedings, 2000. XII, 540 pages. 2000.

Vol. 1778: S. Wermter, R. Sun (Eds.), Hybrid Neural Systems. IX, 403 pages. 2000. (Subseries LNAI).

Vol. 1780: R. Conradi (Ed.), Software Process Technology. Proceedings, 2000. IX, 249 pages. 2000.

Vol. 1781: D.A. Watt (Ed.), Compiler Construction. Proceedings, 2000. X, 295 pages. 2000.

Vol. 1782: G. Smolka (Ed.), Programming Languages and Systems. Proceedings, 2000. XIII, 429 pages. 2000.

Vol. 1783: T. Maibaum (Ed.), Fundamental Approaches to Software Engineering. Proceedings, 2000. XIII, 375 pages. 2000.

Vol. 1784: J. Tiuryn (Eds.), Foundations of Software Science and Computation Structures. Proceedings, 2000. X, 391 pages. 2000.

Vol. 1785: S. Graf, M. Schwartzbach (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. Proceedings, 2000. XIV, 552 pages. 2000.

Vol. 1786: B.H. Haverkort, H.C. Bohnenkamp, C.U. Smith (Eds.), Computer Performance Evaluation. Proceedings, 2000. XIV, 383 pages. 2000.

Vol. 1790: N. Lynch, B.H. Krogh (Eds.), Hybrid Systems: Computation and Control. Proceedings, 2000. XII, 465 pages. 2000.

Vol. 1792: E. Lamma, P. Mello (Eds.), AI*IA 99: Advances in Artificial Intelligence. Proceedings, 1999. XI, 392 pages. 2000. (Subseries LNAI).

Vol. 1793: O. Cairo, L.E. Sucar, F.J. Cantu (Eds.), MICAI 2000: Advances in Artificial Intelligence. Proceedings, 2000. XIV, 750 pages. 2000. (Subseries LNAI).

Vol. 1794: H. Kirchner, C. Ringeissen (Eds.), Frontiers of Combining Systems. Proceedings, 2000. X, 291 pages. 2000. (Subseries LNAI).

Vol. 1795: J. Sventek, G. Coulson (Eds.), Middleware 2000. Proceedings, 2000. XI, 436 pages. 2000.

Vol. 1796: B. Christianson, B. Crispo, J.A. Malcolm, M. Roe (Eds.), Security Protocols. Proceedings, 1999. XII, 229 pages. 2000.

Vol. 1801: J. Miller, A. Thompson, P. Thomson, T.C. Fogarty (Eds.), Evolvable Systems: From Biology to Hardware. Proceedings, 2000. X, 286 pages. 2000.

Vol. 1802: R. Poli, W. Banzhaf, W.B. Langdon, J. Miller, P. Nordin, T.C. Fogarty (Eds.), Genetic Programming. Proceedings, 2000. X, 361 pages. 2000.

Vol. 1803: S. Cagnoni et al. (Eds.), Real-World Applications of Evolutionary Computing. Proceedings, 2000. XII, 396 pages. 2000.

Vol. 1805: T. Terano, H. Liu, A.L.P. Chen (Eds.), Knowledge Discovery and Data Mining. Proceedings, 2000. XIV, 460 pages. 2000. (Subseries LNAI).