Johannes Fürnkranz
Tobias Scheffer
Myra Spiliopoulou (Eds.)

LNAI 4213

Knowledge Discovery in Databases: PKDD 2006

10th European Conference on Principles and Practice of Knowledge Discovery in Databases
Berlin, Germany, September 2006, Proceedings

Springer

Lecture Notes in Artificial Intelligence 4213

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Johannes Fürnkranz Tobias Scheffer
Myra Spiliopoulou (Eds.)

Knowledge Discovery in Databases: PKDD 2006

10th European Conference on Principle and Practice of Knowledge Discovery in Databases
Berlin, Germany, September 18-22, 2006
Proceedings

Springer

Series Editors

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

Volume Editors

Johannes Fürnkranz
Technische Universität Darmstadt
Fachbereich Informatik
Hochschulstraße 10, 64289 Darmstadt, Germany
E-mail: juffi@ke.informatik.tu-darmstadt.de

Tobias Scheffer
Humboldt-Universität zu Berlin
Institut für Informatik
Unter den Linden 6, 10099 Berlin, Germany
E-mail: scheffer@informatik.hu-berlin.de

Myra Spiliopoulou
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Universitätsplatz 2, 39016 Magdeburg, Germany
E-mail: myra@iti.cs.uni-magdeburg.de

Library of Congress Control Number: 2006932511

CR Subject Classification (1998): I.2, H.2, J.1, H.3, G.3, I.7, F.4.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-45374-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-45374-1 Springer 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. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11871637 06/3142 5 4 3 2 1 0

Preface

The two premier annual European conferences in the areas of machine learning and data mining have been collocated ever since the joint conference in Freiburg, Germany, 2001. The European Conference on Machine Learning was established 20 years ago, when the first European Working Session on Learning was held in Orsay, France, in 1986. The conference is growing, and is more lively than ever. The European Conference on Principles and Practice of Knowledge Discovery in Databases celebrates its tenth anniversary; the first PKDD took place in 1997 in Trondheim, Norway. Over the years, the ECML/PKDD series has evolved into one of the largest and most selective international conferences in these areas, the only one that provides a common forum for the two closely related fields. In 2006, the 6th collocated ECML/PKDD took place during September 18-22, when the Humboldt-Universität zu Berlin hosted the 17th European Conference on Machine Learning (ECML) and the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).

The successful model of a hierarchical reviewing process that was introduced last year for the ECML/PKDD 2005 in Porto has been taken over in 2006. We nominated 32 Area Chairs, each of them responsible for several closely related research topics. Suitable areas were selected on the basis of the submission statistics for ECML/PKDD 2005 to ensure a proper load balance among the Area Chairs. For the first time, a joint Program Committee was nominated for the two conferences, consisting of 280 renowned researchers, mostly proposed by the Area Chairs. This joint PC, the largest of the series to date, allowed us to exploit synergies and deal competently with topic overlaps between ECML and PKDD.

ECML/PKDD 2006 received 564 full paper submissions that entered the reviewing process. The submissions were manually assigned to the Area Chairs, who coordinated the reviewers thereafter. Reviewer assignment was based on bidding with CyberChairPRO, as in the previous years. With very few exceptions, every submission was reviewed by three PC members. Based on these reviews, on feedback from the authors, and on discussions among the reviewers, the Area Chairs provided a recommendation for each paper. Continuing the tradition of previous events in the series, we accepted full papers for oral presentation and short papers for poster presentation. The final decision was made by us based on the recommendations of the Area Chairs. We selected 46 full papers and 36 short papers for ECML, and 36 full papers and 26 short papers for PKDD. The acceptance rate for full papers is 14.5% and the overall acceptance rate is 25.5%, in accordance with the high-quality standards of the conference series. Next to the paper and poster sessions, ECML/PKDD 2006 also featured five invited talks, ten workshops, seven tutorials and the ECML/PKDD discovery challenge.

We distinguished eight outstanding contributions; the awards were generously sponsored by the *Machine Learning Journal* and the *KD-Ubiq network*.

ECML Best Paper: Quoc Le, Alex Smola, Thomas Gärtner, Yasemin Altun: *Transductive Gaussian Process Regression with Automatic Model Selection.*
PKDD Best Paper: Pauli Miettinen, Taneli MielikÃď̃en, Aristides Gionis, Gautam Das, Heikki Mannila: *The Discrete Basis Problem.*
ECML Best Student Paper: Bernd Gutmann and Kristian Kersting: *TildeCRF. Conditional Random Fields for Logical Sequences.*
PKDD Best Student Paper: Arik Friedmann, Assaf Schuster, Ran Wolff: *k-Anonymous Decision Tree Induction.*
ECML Innovative Contribution: Alexander Clark, Christophe Costa Florencio, Chris Watkins: *Languages as Hyperplanes: Grammatical Inference with String Kernels.*
PKDD Innovative Application: Herna Viktor, Eric Paquet, Hongyu Guo: *Measuring to Fit: Virtual Tailoring Through Cluster Analysis and Classification.*

The *ECML/PKDD Best Presentation* and the *ECML/PKDD Best Poster Presentation* awards were elected by participants of the conference.

This year's *Discovery Challenge* focused on personalized spam filtering and generalization across related learning tasks. Steffen Bickel organized the challenge; 26 teams participated. For task A, three teams achieved a first rank: Khurram Nazir Junejo, Mirza Muhammad Yousaf, and Asim Karim; Bernhard Pfahringer; and Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, and Chirag Taneja. Task B was won by Gordon Cormack. The solution of Bernhard Pfahringer was distinguished with the Creativity Award.

We are indebted to the Area Chairs, Program Committee members and external reviewers for their effort and engagement in making a rich but selective scientific program for ECML/PKDD. Special thanks go to those reviewers that helped with additional reviews at very short notice to assist with at difficult decisions. We further thank our two workshop and tutorial chairs Tapio Elomaa and Bart Goethals for selecting and coordinating the ten workshop and seven tutorial events that accompanied the conference; the workshop organizers, tutorial presenters, and the organizers of the discovery challenge; Richard van de Stadt and CyberChairPRO for competent and flexible support; the Local Organizing Committee; and all other people that contributed to the organization of this event. Finally, we are grateful to the the Steering Committee and the ECML/PKDD community that entrusted us with the organization of ECML/PKDD 2006.

Most of all, however, we would like to thank all the authors who honored us by submitting their work to this conference, thereby facilitating the success of this event.

September 2006

Johannes Fürnkranz
Tobias Scheffer
Myra Spiliopoulou

Sponsors

We wish to express our gratitude to our sponsors for their great contributions to the conference. We wish to thank Google for featuring the Google ECML Poster Reception and providing ten Student Travel Awards; the Humboldt-Universität zu Berlin for providing the conference venue; the German Science Foundation DFG for supporting all invited speakers; KD-Ubiq for supporting the PKDD Poster Reception and European Projects Poster Reception, four Student Travel Awards, and the Best Paper Awards; the European Office of Aerospace Research and Development (EOARD), Air Force Office of Scientific Research, United States Air Force Research Laboratory for generous financial support; Strato AG for providing the awards to the winners of the Discovery Challenge; the Pascal Network of Excellence and IBM for financial support; the *Machine Learning Journal* for supporting the Student Best Paper Awards.

Google™

HUMBOLDT-UNIVERSITÄT ZU BERLIN

Deutsche Forschungsgemeinschaft
DFG

KD ubiq

EUROPEAN OFFICE OF AEROSPACE R&D

STRATO AG

PASCAL
Pattern Analysis, Statistical Modelling and Computational Learning

IBM

Machine Learning

AFOSR/EOARD support is not intended to express or imply endorsement by the U.S. Federal Government.

Organization

Program Chairs

Johannes Fürnkranz (Technische Universität Darmstadt)
Tobias Scheffer (Humboldt-Universität zu Berlin)
Myra Spiliopoulou (Otto-von-Guericke-Universität Magdeburg)

Local Chairs

Andreas Nürnberger (Otto-von-Guericke-Universität Magdeburg)
Tobias Scheffer (Humboldt-Universität zu Berlin)

Workshop and Tutorial Chairs

Tapio Elomaa (Tampere University of Technology)
Bart Goethals (University of Antwerp)

Discovery Challenge Chair

Steffen Bickel (Humboldt-Universität zu Berlin)

Poster and Fashion Designer, Webmaster

Michael Brückner (Humboldt-Universität zu Berlin)

Local Arrangements Team

Steffen Bickel
Michael Brückner
Thoralf Klein
Ulf Brefeld
Isabel Drost

Steering Committee

Hendrik Blockeel
Pavel Brazdil
Floriana Esposito
Dragan Gamberger
Alípio Jorge
Dino Pedreschi
Luís Torgo
Jean-François Boulicaut
Rui Camacho
João Gama
Fosca Gianotti
Nada Lavrač
Ljupčo Todorovski

Area Chairs

Hendrik Blockeel
Soumen Chakrabarti
Pádraig Cunningham
Kurt Driessens
Peter Flach
Thomas Gärtner
Bart Goethals
George Karypis
Stefan Kramer
Katharina Morik
Bernhard Pfahringer
Carl Rasmussen
Dale Schuurmans
Csaba Szepesvári
Luís Torgo
Vishy Vishwanathan
Henrik Boström
Olivier Chapelle
James Cussens
Ronen Feldman
Eibe Frank
João Gama
Thomas Hofmann
Kristian Kersting
Stan Matwin
Klaus Obermayer
Luc De Raedt
Lorenza Saitta
Jaideep Srivastava
Hannu Toivonen
Volker Tresp
Mohammed Zaki

Program Committee

James Abello
Douglas Aberdeen
Gediminas Adomavicius
Charu Aggarwal
Eugene Agichtein
Jesus S. Aguilar-Ruiz
Erick Alphonse
Yasemin Altun
Massih Amini
Aijun An
Josep-Lluis Arcos
Yonatan Aumann
Paolo Avesani
Antonio Bahamonde
David Barber
Roberto Bayardo
Bettina Berendt
Michael R. Berthold
Elisa Bertino
Fernando Berzal
Steffen Bickel
Francesco Bonchi
Gianluca Bontempi
Remco Bouckaert
Jean-François Boulicaut
Thorsten Brants
Mikio Braun
Pavel Brazdil
Ulf Brefeld
Derek Bridge
Björn Bringmann
Klaus Brinker
Wray Buntine
Christian Böhm
Tiberio Caetano
Toon Calders
Rui Camacho
Barbara Catania
Nicolò Cesa-Bianchi
Deepayan Chakrabarti
Sharma Chakravarthy
LiWu Chang
David Cheung
Amanda Clare
Ian Cloete
Antoine Cornuéjols
Koby Crammer
Susan Craw

Daniel Cremers
Bruno Cremilleux
Walter Daelemans
Ernst Damien
Hal Daume III
Ian Davidson
Luc Dehaspe
Olivier Delalleau
Janez Demšar
François Denis
Chris Ding
Chabane Djeraba
Sašo Džeroski
Tina Eliassi-Rad
Tapio Elomaa
Floriana Esposito
Roberto Esposito
Martin Ester
Wei Fan
Ad Feelders
Arthur Flexer
George Forman
Matthias Franz
Paolo Frasconi
Glenn Fung
Linda C. van der Gaag
Dragan Gamberger
Jean-Gabriel Ganascia
Minos Garofalakis
Floris Geerts
Pierre Geurts
Ali Ghodsi
Joydeep Ghosh
Fosca Giannotti
Aristides Gionis
Attilio Giordana
Christophe Giraud-Carrier
Mark Girolami
David Gondek
Gunter Grieser
Marko Grobelnik
Valerie Guralnik
Mark Hall
Howard Hamilton
Eui-Hong Han
Markus Hegland
Jose Hernandez-Orallo
Gerhard Heyer
Colin de la Higuera
Melanie Hilario
Alexander Hinneburg
Susanne Hoche
Achim Hoffmann
Jaakko Hollmén
John H. Holmes
Geoffrey Holmes
Tamas Horvath
Andreas Hotho
San-Yih Hwang
Frank Höppner
Eyke Hüllermeier
Nitin Indurkhya
Iñaki Inza
Manfred Jaeger
Szymon Jaroszewicz
Edwin de Jong
Alípio Jorge
Alexandros Kalousis
Jaz Kandola
Hillol Kargupta
Andreas Karwath
Samuel Kaski
Motoaki Kawanabe
Eamonn Keogh
Latifur Khan
Ross King
Mika Klemettinen
Ralf Klinkenberg
Arno Knobbe
Jens Kohlmorgen
Igor Kononenko
Adam Kowalczyk
Miroslav Kubat
Nicolas Lachiche
Michail Lagoudakis
Pedro Larrañaga
Pavel Laskov
Mark Last
Dominique Laurent
Nada Lavrač

Ulf Leser
Jure Leskovec
Christina Leslie
Ee-Peng Lim
Bing Liu
John Lloyd
Huma Lodhi
Jose A. Lozano
Ulrike von Luxburg
Donato Malerba
Suresh Manandhar
Giuseppe Manco
Heikki Mannila
Yannis Manolopoulos
Dragos Margineantu
Yuji Matsumoto
Michael May
Vasileios Megalooikonomou
Wagner Meira Jr.
Prem Melville
Rosa Meo
Taneli Mielikäinen
Rada Mihalcea
Brian Milch
Dunja Mladenic
Bamshad Mobasher
Klaus-Robert Müller
Hector Muñoz-Avila
Alexandros Nanopoulos
Olfa Nasraoui
Claire Nedellec
Kee Siong Ng
Siegfried Nijssen
Mahesan Niranjan
Richard Nock
Ann Nowe
Andreas Nürnberger
Arlindo Oliveira
Manfred Opper
Carlos Ordonez
Miles Osborne
Martijn van Otterlo
Gerhard Paaß
Balaji Padmanabhan
Georgios Paliouras
Spiros Papadimitriou
Srinivasan Parthasarathy
Andrea Passerini
Dino Pedreschi
Jian Pei
Lourdes Peña
Jean-Marc Petit
Johann Petrak
Enric Plaza
William M. Pottenger
Doina Precup
Stephen Pulman
Joaquin Quiñonero-Candela
Naren Ramakrishnan
Jan Ramon
Rajeev Rastogi
Francesco Ricci
Christophe Rigotti
Gilbert Ritschard
Céline Robardet
Marko Robnik-Sikonja
Romer Rosales
Volker Roth
Juho Rousu
Céline Rouveirol
Stefan Rüping
Ulrich Rückert
Cordelia Schmid
Martin Scholz
Nic Schraudolph
Anton Schwaighofer
Michele Sebag
Marc Sebban
Matthias Seeger
Bernhard Seeger
Giovanni Semeraro
Petteri Sevon
Ayumi Shinohara
Arno Siebes
Dan Simovici
Neil Smalheiser
Padhraic Smyth
Carlos Soares
Stephen Soderland
Maarten van Someren

Alessandro Sperduti
Eduardo Spinosa
Nicolas Spyratos
Michael Steinbach
Jan Struyf
Gerd Stumme
Einoshin Suzuki
Vojtech Svatek
Marcin Szczuka
Prasad Tadepalli
Pang-Ning Tan
Tao Tao
Takao Terano
Henry Tirri
Ljupco Todorovski
Ioannis Tsochantaridis
Karl Tuyls
Michalis Vazirgiannis
Katja Verbeeck
Jean-Philippe Vert
Rene Vidal
Jianyong Wang
Wei Wang
Ke Wang
Jason Wang
Haixun Wang
Takashi Washio
Louis Wehenkel
Sholom Weiss
Gerhard Widmer
Marco Wiering
Eric Wiewiora
Ole Winther
Nirmalie Wiratunga
Stefan Wrobel
Hui Xiong
Jiong Yang
Ying Yang
Philip Yu
Kai Yu
Bianca Zadrozny
Gerson Zaverucha
Filip Zelezny
Thomas Zeugmann
ChengXiang Zhai
Byoung-Tak Zhang
Ying Zhao
Dengyong Zhou
Xiaojin Zhu
Alexander Zien
Jean-Daniel Zucker
Blaž Zupan

Additional Reviewers

Sreangsu Acharyya
Elke Achtert
Raman Adaikkalavan
Fabio Aiolli
Florence d'Alché-Buc
Claudia d'Amato
Bill Andreopoulos
Annalisa Appice
Eva Armengol
Stella Asiimwe
Anneleen Van Assche
Maurizio Atzori
Korinna Bade
Gökhan H. Bakır
Pierpaolo Basile
Matthew Beal
Gurkan Bebek
Aditya Belapukar
Margherita Berardi
Jürgen Beringer
Marc Bernard
Kanishka Bhaduri
Marenglen Biba
Hanczar Blaise
Gilles Blanchard
Yann-ael Le Borgne
Karsten Borgwardt
Marco Botta
Dominique Bouthinon
Janez Brank

Michael Brückner
Robert D. Burbidge
Emma Byrne
Nicolas Cebron
Michelangelo Ceci
Eugenio Cesario
Hasan E. Cetingul
Sutanu Chakraborti
Hari Chelladurai
Min Chen
Haibin Cheng
Hong Cheng
Li Cheng
Yun Chi
Silvia Chiappa
Thiago Cordeiro
Gianni Costa
Marco Cristo
Tom Croonenborghs
Vassilis Cutsuridis
Wisam Dakka
Souptik Datta
Sandra De Amo
Fabien De Marchi
Marco Degemmis
Evangelos Dellis
Nicola Di Mauro
Isabel Drost
Norberto Díaz Díaz
Lauri Eronen
Vicent Estruch
Nicola Fanizzi
Francisco J. Ferrer-Troyano
Stefano Ferilli
Cèsar Ferri
Frédéric Flouvat
Francesco Folino
Nuno Fonseca
Blaz Fortuna
Murat Ganiz
Jing Gao
Dimitrios Gavrilis
Peter Vincent Gehler
Liqiang Geng
Pierre Gerard
Atiyeh Ghoreyshi
Amol Ghoting
Arnaud Giacometti
Chris Giannella
Aris Gkoulalas-Divanis
Robby Goetschalckx
Alvina Goh
Ricardo Grau
Steve Greenwald
Gunjan Gupta
Simon Günter
Benjamin Haibe-Kains
Maria Halkidi
Hongxing He
Matthias Hein
Christoph Heinz
Corneliu Henegar
Katherine G. Herbert
Kirsten Hildrum
Petteri Hintsanen
Wai-Shing Ho
Meng Hu
Ming Hua
Jiayuan Huang
Yi Huang
Alex Ihler
Jaikrishna
Christopher D. Janneck
Robert Jäschke
Tao-Yuan Jen
Antonio Jimeno
Frederic Jurie
Dmitry Kamenetsky
Murat Kantarcioglu
Ioannis Karydis
Balakumar Kendai
Wolf Kienzle
David H. Kim
Seyoung Kim
Jee-Hyub Kim
Kwang In Kim
Jiří Kléma
Ioannis Klafaptis
Christine Körner
Ville Kononen

Stasinos Konstantopoulos
Despina Kontos
Arne Koopman
Petra Kralj
Chase Krumpelman
Jussi Kujala
Matjaž Kukar
Peer Kröger
Peter Kunath
Niels Landwehr
Maggie Man Ki Lau
Anne Laurent
Quoc V Le
Steven Lemm
Shenzhi Li
Ting Li
Oriana Licchelli
Marina Litvak
Alexander Liu
Kun Liu
Yang Liu
George Loizou
Pasquale Lops
Alicia Troncoso Lora
Marc Q. Ma
Patrick Marcel
Keith Marsolo
Eric Martin
Elio Masciari
Stewart Massie
Giuseppe M. Mazzeo
Oscar Meruvia Pastor
Kapila Moonesinghe
Kawanabe Motoaki
Fernando Mourão
Rahman Mukras
Mirco Nanni
Jan Nemrava
Isabel A. Nepomuceno-Chamorro
David Newman
Radu Stefan Niculescu
Blaž Novak
Cheng Soon Ong
Amandine Orecchioni
Matthew Eric Otey
Aritz Pérez
Ignazio Palmisano
Elias Pampalk
Maarten Peeters
Jaakko Peltonen
Michael Perrow
Sergios Petridis
Viet Phan-Luong
Tadek Pietraszek
Aloisio Carlos de Pina
Conrad Plake
Claudia Plant
Mannes Poel
Subhesh Pradhan
Pavel Praks
Kai Puolamäki
Gunnar Rätsch
M. Jose Ramirez-Quintana
S. S. Ravi
Avinash Ravichandran
Simon Rawles
Matthias Renz
Chiara Renso
Konrad Rieck
Stefan Riezler
Carsten Riggelsen
Salvatore Rinzivillo
Pedro Rodrigues
Domingo S. Rodríguez-Baena
Thierson Rosa
Michel de Rougemont
Vishesh Sahu
Luka Šajn
Sagar Savla
Leander Schietgat
Christoph Schmitz
Timon Schroeter
Matthias Schubert
Jerry Scripps
Tomi Silander
Milan Simunek
Dheeraj Singaraju
Ksenia Shalonova
Kazumi Slott
Diego Sona

Bin Song
Le Song
Xiaodan Song
Soeren Sonnenburg
Arnaud Soulet
Junilda Spirollari
Harald Steck
Sebastian Stober
Petr Strossa
Jimeng Sun
Peter Sunehag
Aditya Telang
Choon Hui Teo
Ivan Titov
Igor Trajkovski
George Tsatsaronis
Mihalis Tsoukalos
Miroslav Vacura
Hamed Valizadegan
Antonio Varlaro
Sriharsha Veeramachaneni
Arvind Venkatachalam
Jarkko Venna
Jakob Verbeek
Vassilis Verykios
Akrivi Vlachou
Peter Vrancx
Christian James Walder
Qian Wan
Chao Wang
Lei Wang
Qiang Wang
Xuanhui Wang
Li Wei
Joost van de Weijer
Dietrich Wettschereck
Adam Woznica
Junjie Wu
Mingrui Wu
Xiaopeng Xi
Keisuke Yamazaki
Hui Yang
Xiaoning Yang
Dragomir Yankov
Jin Yu
Shipeng Yu
Monika Zakova
Shijie Zhang
Xinhua Zhang
Bin Zhou
Xingquan Zhu
Arthur Zimek

Table of Contents

Invited Talks

Long Papers

Short Papers

On Temporal Evolution in Data Streams

Charu C. Aggarwal

IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
charu@us.ibm.com

Abstract. In recent years, the progress in hardware technology has made it possible for organizations to store and record large streams of transactional data. This results in databases which grow without limit at a rapid rate. This data can often show important changes in trends over time. In such cases, it is useful to understand, visualize, and diagnose the evolution of these trends. In this talk, we discuss a method to diagnose the changes in the underlying data stream and other related methods for change detection in streams. We also discuss the problem of data stream evolution in the context of mining algorithms such as clustering and classification. In many cases, mining algorithms may not function as effectively because of the change in the underlying data. We discuss the effects of evolution on mining and synopsis construction algorithms and a number of opportunities which may be available for further research on the topic.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, p. 1, 2006.
© Springer-Verlag Berlin Heidelberg 2006

The Future of CiteSeer: CiteSeer^x

C. Lee Giles

David Reese Professor, School of Information Sciences and Technology
Professor, Computer Science and Engineering
Professor, Supply Chain and Information Systems
The Pennsylvania State University
University Park, PA, USA
giles@ist.psu.edu

Abstract. CiteSeer, a public online computer and information science search engine and digital library, was introduced in 1997 and was a radical departure from the traditional methods of academic and scientific document access and analysis. Computer and information scientists quickly became used to and expected immediate access to their literature and CiteSeer provided a popular partial solution. CiteSeer was based on these features: actively acquiring new documents, automatic citation indexing, and automatic linking of citations and documents. CiteSeer, now hosted at the Pennsylvania State University with several mirrors, has over 750,000 documents. The current CiteSeer model is to a limited extent portable and was recently extended to academic business documents (SMEALSearch).

Why has CiteSeer been so popular and how should it progress? What is its role with regards to other similar systems such as the Google Scholar and DBLP? What role should CiteSeer play in the open access movement? We discuss this and the Next Generation CiteSeer project, CiteSeerx, which will emphasize CiteSeer as a research tool, research web service, and researcher facilitator and testbed. In contrast to the current tightly integrated CiteSeer architecture, CiteSeerx will be modular, scalable and self managed. We will discuss how new intelligent data mining and information extraction algorithms will provide improved and new indexes, enhanced document access, expanded and automatic document gathering, collaboratories, new data and metadata resources, active mirroring, and web services. As an example of new features, we point out our new API based acknowledgement index and search. This new feature not only provides insight into the impact of acknowledged individuals, funding agencies and others, but also presents an architectural model for integration and expansion of our legacy system.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, p. 2, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Learning to Have Fun

Jonathan Schaeffer

University of Alberta
Department of Computing Science
jonathan@cs.ualberta.ca

Abstract. Games have played a major role in the history of artificial intelligence research. The goal of this research largely has been to build programs capable of defeating strong human players. Most of the literature has been devoted to two-player, perfect information games—games where the research results have little wide-spread applicability. However, over the past few years the need for improved AI techniques have become apparent in commercial computer games, a $25 billion industry. Laird and van Lent call the new generation of commercial games "AI's killer application". The buying public wants to see realistic artificial intelligence in these products. Here the the metric is a "fun" experience, not winning. Hence, the outcomes from research using these applications will be of much wider applicability. This talk will discuss the challenges of using machine learning in commercial computer games to create "fun".

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, p. 3, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Winning the DARPA Grand Challenge

Sebastian Thrun

Stanford University, Robotics Lab
thrun@stanford.edu

Abstract. The DARPA Grand Challenge has been the most significant challenge to the mobile robotics community in more than a decade. The challenge was to build an autonomous robot capable of traversing 132 miles of unrehearsed desert terrain in less than 10 hours. In 2004, the best robot only made 7.3 miles. In 2005, Stanford won the challenge and the $2M prize money by successfully traversing the course in less than 7 hours. This talk, delivered by the leader of the Stanford Racing Team, will provide insights in the software architecture of Stanford's winning robot. The robot massively relied on machine learning and probabilistic modeling for sensor interpretation, and robot motion planning algorithms for vehicle guidance and control. The speaker will explain some of the basic algorithms and share some of the excitement characterizing this historic event. He will also discuss the implications of this work for the future of the transportation.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, p. 4, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Challenges of Urban Sensing

Henry Tirri

Nokia Research Center
Henry.Tirri@nokia.com

Abstract. Wireless sensor networks are emerging as a critical information technology, and they are continuing the trend originating in mainframe computing currently at the stage of mobile computing. This trend shows several aspects consistent in the evolution of computing including the increasing hardware miniaturization of the computing units and an increasing emphasis of the role of communication between the computing units – "networking". In addition from the software side there is an increasing need to software solutions that are robust, exhibit distributed control, collaborative interfaces resulting in adaptive capabilities also at the system level. Like the present Internet, wireless sensor networks are large-scale distributed systems, but composed of smart sensors and actuators. They will eventually infuse the physical world and provide "grounding" for the Internet thus creating the Internet of Things. Research on wireless sensor networks has been taking place at several levels, from the lowest physical level to the highest information level – the latter is much less developed than the research at the physical levels. In addition, much of the research in wireless sensor networks has been focusing on military or science applications. However, wireless sensor networks can also play an important role in the realization of ubiquitous computing for everyday life - creating what we call "Urban sensing environment". In urban sensing many natural gateways exist to collect and process the sensor information – static ones such as media devices, or mobile devices such as smart phones that can collect sensor information when entering the communication range of an active sensor. Some of the applications of wireless sensor network technology at home include, in addition to the surveillance functions, adding "intelligence" to utility consumption, electronic tagging, contamination control and disaster monitoring. Similarly at the community level "traffic monitoring" including people allows a development of totally unseen services from micro weather forecasts to new ways for "sensing the environment" for entertainment. In this talk we will outline some of the research challenges for urban sensing, and the role of learning and data analysis techniques for solving those challenges.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, p. 5, 2006.
© Springer-Verlag Berlin Heidelberg 2006

SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery

Martin Atzmueller and Frank Puppe

University of Würzburg, 97074 Würzburg, Germany
Department of Computer Science
Tel.: +49 931 888-6739; Fax: +49 931 888-6732
{atzmueller, puppe}@informatik.uni-wuerzburg.de

Abstract. In this paper we present the novel SD-Map algorithm for exhaustive but efficient subgroup discovery. SD-Map guarantees to identify all interesting subgroup patterns contained in a data set, in contrast to heuristic or sampling-based methods. The SD-Map algorithm utilizes the well-known FP-growth method for mining association rules with adaptations for the subgroup discovery task. We show how SD-Map can handle missing values, and provide an experimental evaluation of the performance of the algorithm using synthetic data.

1 Introduction

Subgroup discovery [1,2,3,4] is a powerful and broadly applicable method that focuses on discovering interesting subgroups of individuals. It is usually applied for data exploration and descriptive induction, in order to identify relations between a dependent (target) variable and usually many independent variables, e.g., "the subgroup of 16-25 year old men that own a sports car are more likely to pay high insurance rates than the people in the general population", but can also be used for classification (e.g., [5]).

Due to the exponential search space commonly heuristic (e.g., [2,6]) or sampling approaches (e.g., [7]) are applied in order to discover the k best subgroups. However, these do not guarantee the discovery of all the optimal (most interesting) patterns. For example, if a heuristic method implements a greedy approach or uses certain pruning heuristics, then whole subspaces of the search space are not considered. Furthermore, heuristic approaches usually have "blind spots" that prevent certain patterns from being discovered: Considering beam search for example, if only a combination of factors is interesting, where each single factor is not, then the combination might not be identified [2]. In consequence, often not all interesting patterns can be discovered by the heuristic methods, and only estimates about the best k subgroups are obtained. This can be a problem especially for explorative and descriptive methods since some of the truly interesting patterns cannot be identified. In contrast to heuristic methods exhaustive approaches guarantee to discover the best solutions. However, then the runtime costs of an exhaustive algorithm usually prohibit its application for larger search spaces.

In this paper we propose the novel *SD-Map* algorithm for fast and exhaustive subgroup discovery, based upon the FP-growth algorithm [8] for mining association rules with adaptations for the subgroup discovery task. We show how SD-Map can efficiently

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 6–17, 2006.
© Springer-Verlag Berlin Heidelberg 2006

handle missing values that are common for some domains, and also how the algorithm can be applied for more complex description languages using internal disjunctions. Additionally, we provide an experimental evaluation of the algorithm using synthetic data.

The rest of the paper is organized as follows: We introduce the subgroup discovery setting in Section 2, and present the SD-Map algorithm in Section 3. After that, an experimental evaluation of the algorithm using synthetic data is provided in Section 4. Finally, we conclude the paper with a discussion of the presented work, and we show promising directions for future work in Section 5.

2 Subgroup Discovery

The main application areas of subgroup discovery are exploration and descriptive induction, to obtain an overview of the relations between a target variable and a set of explaining variables. Then, not necessarily complete relations but also partial relations, i.e., (small) subgroups with "interesting" characteristics can be sufficient.

Before describing the subgroup discovery setting, we first introduce the necessary notions concerning the used knowledge representation: Let Ω_A be the set of all attributes. For each attribute $a \in \Omega_A$ a range $dom(a)$ of values is defined. Furthermore, we assume $\mathcal{V}_A$ to be the (universal) set of attribute values of the form $(a = v)$, where $a \in \Omega_A$ is an attribute and $v \in dom(a)$ is an assignable value. We consider nominal attributes only so that numeric attributes need to be discretized accordingly. Let CB be the case base (data set) containing all available cases, also often called instances. A case $c \in CB$ is given by the n-tuple $c = ((a_1 = v_1), (a_2 = v_2), \ldots, (a_n = v_n))$ of $n = |\Omega_A|$ attribute values, where $v_i \in dom(a_i)$ for each a_i.

A subgroup discovery setting mainly relies on the following four main properties: the target variable, the subgroup description language, the quality function, and the search strategy. In this paper we focus on binary target variables. Similar to the *MIDOS* approach [3] we try to identify subgroups that are, e.g., as large as possible, and have the most unusual (distributional) characteristics with respect to the concept of interest given by the target variable.

The description language specifies the individuals belonging to the subgroup. In the case of single-relational propositional languages a subgroup description can be defined as follows:

Definition 1 (Subgroup Description). *A subgroup description $sd = \{e_1, e_2, \ldots, e_n\}$ is defined by the conjunction of a set of selection expressions. These selectors $e_i = (a_i, V_i)$ are selections on domains of attributes, $a_i \in \Omega_A, V_i \subseteq dom(a_i)$. We define Ω_{sd} as the set of all possible subgroup descriptions.*

A quality function measures the interestingness of the subgroup mainly based on a statistical evaluation function, e.g., the chi-squared statistical test. It is used by the search method to rank the discovered subgroups during search. Typical quality criteria include the difference in the distribution of the target variable concerning the subgroup and the general population, and the subgroup size. We assume that the user specifies a minimum support threshold $\mathcal{T}_{Supp}$ corresponding to the number of target class cases contained in the subgroup in order to prune very small and therefore uninteresting subgroup patterns.

Definition 2 (Quality Function). *Given a particular target variable $t \in \mathcal{V}_A$, a quality function $q : \Omega_{sd} \times \mathcal{V}_A \rightarrow R$ is used in order to evaluate a subgroup description $sd \in \Omega_{sd}$, and to rank the discovered subgroups during search.*

Several quality functions were proposed, e.g., [1,2,7]. For binary target variables, examples for quality functions are given by

$$q_{BT} = \frac{(p - p_0) \cdot \sqrt{n}}{\sqrt{p_0 \cdot (1 - p_0)}} \cdot \sqrt{\frac{N}{N - n}}, \quad q_{RG} = \frac{p - p_0}{p_0 \cdot (1 - p_0)},$$

where p is the relative frequency of the target variable in the subgroup, p_0 is the relative frequency of the target variable in the total population, $N = |CB|$ is the size of the total population, and n denotes the size of the subgroup.

Considering the subgroup search strategy an efficient search is necessary, since the search space is exponential concerning all the possible selectors of a subgroup description. In the next section, we propose the novel SD-Map algorithm for exhaustive subgroup discovery based on the *FP-growth* [8] method.

3 SD-Map

SD-Map is an exhaustive search method, dependent on a minimum support threshold: If we set the minimum support to zero, then the algorithm performs an exhaustive search covering the whole unpruned search space. We first introduce the basics of the FP-growth method, referring to Han et al. [8] for a detailed description. We then discuss the SD-Map algorithm and describe the extensions and adaptations of the FP-growth method for the subgroup discovery task. After that, we describe how SD-Map can be applied efficiently for the special case of (strictly) conjunctive languages using internal disjunctions, and discuss related work.

3.1 The Basic FP-Growth Algorithm

The FP-growth [8] algorithm is an efficient approach for mining frequent patterns. Similar to the Apriori algorithm [9], FP-growth operates on a set of items which are given by a set of selectors in our context. The main improvement of the FP-growth method compared to Apriori is the feature of avoiding multiple scans of the database for testing each frequent pattern. Instead, a recursive divide-and-conquer technique is applied.

As a special data structure, the frequent pattern tree or FP-tree is implemented as an extended prefix-tree-structure that stores count information about the frequent patterns. Each node in the tree is a tuple (selector, count, node-link): The count measures the number of times the respective selector is contained in cases reached by this path, and the node-link links to the nodes in the FP-tree with the same assigned selector. The construction of an FP-tree only needs two passes through the set of cases: In the first pass the case base is scanned collecting the set of frequent selectors L that is sorted in support descending order. In the second pass, for each case of the case base the contained frequent selectors are inserted into the tree according to the order of L, such that the chances of sharing common prefixes (subsets of selectors) are increased. After

construction, the paths of the tree contain the count information of the frequent selectors (and patterns), usually in a much more compact form than the case base.

For determining the set of frequent patterns, the FP-growth algorithm applies a divide and conquer method, first mining frequent patterns containing one selector and then recursively mining patterns conditioned on the occurrence of a (prefix) 1-selector. For the recursive step, a conditional FP-tree is constructed, given the conditional pattern base of a frequent selector of the FP-Tree and its corresponding nodes in the tree. The conditional pattern base consists of all the prefix paths of such a node v, i.e., considering all the paths P_v that node v participates in. Given the conditional pattern base, a (smaller) FP-tree is generated, the conditional FP-tree of v utilizing the adapted frequency counts of the nodes. If the conditional FP-Tree just consists of one path, then the frequent patterns can be generated by considering all the combinations of the nodes contained in the path. Otherwise, the process is performed recursively.

3.2 The SD-Map Algorithm

Compared to association rules that measure the confidence ($precision$) and the $support$ of rules [9], subgroup discovery uses a special quality function to measure the interestingness of a subgroup. A naive adaptation of the FP-growth algorithm for subgroup discovery just uses the FP-growth method to collect the frequent patterns; then we also need to test the patterns using the quality function.

For the subgroup quality computation mainly four parameters are used: The true positives tp (cases containing the target variable t in the given subgroup s), the false positives fp (cases not containing the target t in the subgroup s), and the positives TP and negatives FP regarding the target variable t in the general population of size N. Thus, we could just apply the FP-growth method as is, and compute the subgroup parameters as

- $n = count(s)$,
- $tp = support(s) = count(s \wedge t)$,
- $fp = n - tp$,
- $p = tp/(tp + fp)$, and
- $TP = count(t)$,
- $FP = N - TP$,
- $p_0 = TP/(TP + FP)$.

However, one problem encountered in data mining remains: The missing value problem. If missing values are present in the case base, then not all cases may have a defined value for each attribute, e.g., if the attribute value has not been recorded. For association rule mining this usually does not occur, since we only consider items in a transaction: An item is present or not, and never undefined or missing. In contrast missing values are a significant problem for subgroup mining in some domains, e.g., in the medical domain [4,10]. If the missing values cannot be eliminated they need to be considered in the subgroup discovery method when computing the quality of a subgroup: We basically need to adjust the counts for the population by identifying the cases where the subgroup variables or the target are not defined, i.e., where these have missing values.

So, the simple approach described above is redundant and also not sufficient, since (a) we would get a larger tree if we used a normal node for the target; (b) if there are missing values, then we need to restrict the parameters TP and FP to the cases for which all the attributes of the selectors contained in the subgroup description have defined values; (c) furthermore, if we derived $fp = n - tp$, then we could not distinguish the cases where the target is not defined.

In order to improve this situation, and to reduce the size of the constructed FP-tree, we consider the following important observations:

1. For estimating the subgroup quality we only need to determine the four basic subgroup parameters tp, fp, TP, and FP with a potential adaptation for missing values. Then, the other parameters can be derived: $n = tp + fp$, $N = TP + FP$.
2. For subgroup discovery, the concept of interest, i.e., the target variable is fixed, in contrast to the arbitrary "rule head" of association rules. Thus, the necessary parameters described above can be directly computed, e.g., considering the true positives tp, if a case contains the target and a particular selector, then the tp count of the respective node of the FP-tree is incremented.

If the tp, fp counts are stored in the nodes of the FP-tree, then we can compute the quality of the subgroups directly while generating the frequent patterns. Furthermore, we only need to create nodes for the independent variables, not for the dependent (target) variable. In the SD-Map algorithm, we just count for each node if the target variable occurs (incrementing tp) or not (incrementing fp), restricted to cases for which the target variable has a defined value. The counts in the general population can then be acquired as a by-product.

For handling missing values we propose to construct a second FP-tree-structure, the *Missing-FP-tree*. The FP-tree for counting the missing values can be restricted to the set of frequent attributes of the main FP-Tree, since only these can form subgroup descriptions that later need to be checked with respect to missing values. Then, the Missing-FP-tree needs to be evaluated in a special way to obtain the respective missing counts. To adjust for missing values we only need to adjust the number of TP, FP corresponding to the population, since the counts for the subgroup FP-tree were obtained for cases where the target was defined, and the subgroup description only takes cases into account where the selectors are defined. Using the Missing-FP-tree, we can identify the situations where any of the attributes contained in the subgroup description is undefined: For a subgroup description $sd = (e_1 \wedge e_2 \wedge \cdots \wedge e_n), e_i = (a_i, V_i), V_i \subseteq dom(a_i)$ we need to compute the missing counts $missing(a_1 \vee a_2 \vee \cdots \vee a_n)$ for the set of attributes $M = \{a_1, \ldots, a_n\}$. This can be obtained applying the following transformation:

$$missing(a_1 \vee \cdots \vee a_n) = \sum_{i=1}^{n} missing(a_i) - \sum_{m \in 2^M} missing(m)\,, \qquad (1)$$

where $|m| \geq 2$. Thus, in order to obtain the *missing adjustment* with respect to the set M containing the attributes of the subgroup, we need to add the entries of the header nodes of the Missing-FP-tree corresponding to the individual attributes, and subtract the entries of every suffix path ending in an element of M that contains at least another element of M.

Algorithm 1. SD-Map algorithm

Require: target variable t, quality function q, set of selectors E (search space)

1: Scan 1 – Collect the frequent set of selectors, and construct the frequent node list L for the main FP-tree:
   1. For each case for which the target variable has a defined value, count the (tp_e, fp_e) for each selector $e \in E$.
   2. Prune all selectors which are below a minimum support $\mathcal{T}_{Supp}$, i.e., $tp(e) = frequency(e \wedge t) < \mathcal{T}_{Supp}$.
   3. Using the unpruned selectors $e \in E$, construct and sort the frequent node list L in support/frequency descending order.
2: Scan 2 – Build the main FP-tree:
   For each node contained in the frequent node list, insert the node into the FP-tree (according to the order of L), if observed in a case, and count the number of (tp, fp) for each node
3: Scan 3 – Build the Missing-FP-tree (This step can also be integrated as a logical step into scan 2):
   1. For all frequent attributes, i.e., the attributes contained in the frequent nodes L of the FP-tree, generate a node denoting the missing value for this attribute.
   2. Then, construct the Missing-FP-tree, counting the (tp, fp) for each (attribute) node.
4: Perform the adapted FP-growth method to generate subgroup patterns:
5: **repeat**
6: **for** each subgroup s_i that denotes a frequent subgroup pattern **do**
7: Compute the adjusted population counts TP , FP as shown in Equation 2
8: Given the parameters tp, fp, TP , FP compute the subgroup quality $q(s_i)$ using a quality function q
9: **until** FP-growth is finished
10: Post-process the set of the obtained subgroup patterns, e.g., return the k best subgroups, or return the subgroup set $S = \{s \mid q(s) \geq q_{min}\}$, for a minimal quality threshold $q_{min} \in \mathbb{R}$ (This step can also be integrated as a logical filtering step in the discovery loop (lines 5-9)).

By considering the (tp, fp) counts contained in the Missing-FP-tree we obtain the number of $(TP_{missing}, FP_{missing})$ where the cases cannot be evaluated statistically since any of the subgroup variables are not defined, i.e., at least one attribute contained in M is missing. To adjust the counts of the population, we compute the correct counts as follows:

$$TP' = TP - TP_{missing}, FP' = FP - FP_{missing}\,. \tag{2}$$

Then, we can compute the subgroup quality based on the parameters tp, fp, TP', FP'. Thus, in contrast to the standard FP-growth method, we do not only compute the frequent patterns in the FP-growth algorithm, but we can also directly compute the quality of the frequent subgroup patterns, since all the parameters can be obtained in the FP-growth step. So, we perform an integrated grow-and-test step, accumulating the subgroups directly. The SD-Map algorithm is shown in Algorithm 1.

SD-Map includes a post-processing step for the selection and the potential redundancy management of the obtained set of subgroups. Usually, the user is interested only in the best k subgroups and does not want to inspect all the discovered subgroups. Thus, we could just select the best k subgroups according to the quality function and return these. Alternatively, we could also choose all the subgroups above a minimum quality

threshold. Furthermore, in order to reduce overlapping and thus potentially redundant subgroups, we can apply post-processing, e.g., clustering methods or a weighted covering approach (e.g., [11]), as in the Apriori-SD [12] algorithm. The post-processing step could potentially also be integrated into the "discovery loop" (while applying the adapted FP-growth method), see Algorithm 1.

A subgroup description as defined in Section 2 can either contain selectors with internal disjunctions, or not. Using a subgroup description language without internal disjunctions is often sufficient for many domains, e.g., for the medical domain [4,10]. In this case the description language matches the setting of the common association rule mining methods. If internal disjunctions are not required in the subgroup descriptions, then the SD-Map method can be applied in a straight-forward manner: If we construct a selector $e = (a, \{v_i\})$ for each value v_i of an attribute a, then the SD-Map method just derives the desired subgroup descriptions: Since the selectors do not overlap, a conjunction of selectors for the same attribute results in an empty set of covered cases. Thus, only conjunctions of selectors will be regarded as interesting if these correspond to a disjoint set of attributes. Furthermore, each path of a constructed frequent pattern tree will also only contain a set of selectors belonging to a disjoint set of attributes.

If internal disjunctions are required, then the search space is significantly enlarged in general, since there are $2^m - 1$ (non-empty) value combinations for an attribute with m values. However, the algorithm can also be applied efficiently for the special case of a description language using selectors with internal disjunctions. In the following section we describe two approaches for handling that situation.

3.3 Applying SD-Map for Subgroup Descriptions with Internal Disjunctions Efficiently

Naive Approach. First, we can just consider all possible selectors with internal disjunctions for a particular attribute. This technique can also be applied if not all internal disjunctions are required, and if only a selection of aggregated values should be used. For example, a subset of the value combinations can be defined by a taxonomy, by the ordinality of an attribute, or by background knowledge, e.g., [4].

If all possible disjunctions need to be considered, then it is easy to see that adding the selectors corresponding to all internal disjunctions significantly extends the set of selectors that are represented in the paths of the tree compared to a subgroup description language without internal disjunctions. Additionally, the selectors overlap: Then, the sizes of many conditional trees being constructed during the mining phase is increased. However, in order to improve the efficiency of this approach we can apply the following pruning techniques:

1. During construction of a conditional tree, we can prune all parent nodes that contain the same attribute as the conditioning node, since these are subsumed by the conditioning selector.
2. When constructing the combinations of the selectors on a single path of the FP-tree we only need to consider the combinations for disjoint sets of the corresponding attributes.

Using Negated Selectors. An alternative approach is especially suitable if all internal disjunctions of an attribute are required. The key idea is to express an internal disjunction by a conjunction of negated selectors: For example, consider the attribute a with the values v_1, v_2, v_3, v_4; instead of the disjunctive selector $(a, \{v_1, v_2\})$ corresponding to the value $v_1 \vee v_2$ we can utilize the conjunctive expression $\neg(a, \{v_3\}) \wedge \neg(a, \{v_4\})$ corresponding to $\neg v_3 \wedge \neg v_4$.

This works for data sets that do not contain missing values. If missing values are present, then we just need to exclude the missing values for such a negated selector: For the negated selectors of the example above, we create $\neg(a, \{v_3, v_{missing}\}) \wedge \neg(a, \{v_4, v_{missing}\})$. Then, the missing values are not counted for the negated selectors. If we need to consider the disjunction of the all attribute values, then we add a selector $\neg(a, \{v_{missing}\})$ for each attribute a. Applying this approach, we only need to create negated selectors for each attribute value of an attribute, instead of adding all internal disjunctions: The set of selectors that are contained in the frequent pattern tree is then significantly reduced compared to the naive approach described above. The evaluation of the negated selectors can be performed without any modifications of the algorithm. Before the subgroup patterns are returned, they are transformed to subgroup descriptions without negation by merging and replacing the negated selectors by semantically equivalent selectors containing internal disjunctions.

3.4 Discussion

Using association rules for classification has already been proposed, e.g., in the *CBA* [13], the *CorClass* [14], and the *Apriori-C* [15] algorithms. Based upon the latter, the Apriori-SD [12] algorithm is an adaptation for the subgroup discovery task. Applying the notion of *class association rules*, all of the mentioned algorithms focus on association rules containing exactly one selector in the rule head.

Compared to the existing approaches, we use the exhaustive FP-growth method that is usually faster than the Apriori approach. To the best of the authors' knowledge, it is the first time that an (adapted) FP-growth method has been applied for subgroup discovery. The adaptations of the Apriori-style methods are also valid for the FP-growth method. Then, due the subgroup discovery setting the memory and runtime complexity of FP-growth can usually be reduced (c.f., [15,12]). In comparison to the Apriori-based methods and a naive application of the FP-growth algorithm, the SD-Map method utilizes a modified FP-growth step that can compute the subgroup quality directly without referring to other intermediate results. If the subgroup selection step is included as a logical filtering step, then the memory complexity can also be decreased even further.

Moreover, none of the existing algorithms tackles the problem of missing values explicitly. Therefore, we propose an efficient integrated method for handling missing values. Such an approach is essential for data sets containing very many missing values. Without applying such a technique, an efficient evaluation of the quality of a subgroup is not possible, since either too many subgroups would need to be considered in a post-processing step, or subgroups with a high quality might be erroneously excluded.

Furthermore, in contrast to algorithms that apply branch-and-bound techniques requiring special (convex) quality functions, e.g., the Cluster-Grouping [16] and the CorClass [14] algorithms, SD-Map can utilize arbitrary quality functions.

4 Evaluation

In our evaluation we apply a data generator for synthetic data that uses a Bayesian network as its knowledge representation. The prior-sample algorithm is then applied in order to generate data sets with the same characteristics, but different sizes. The Bayesian network used for data generation is shown in Figure 1. It models an artificial vehicle insurance domain containing 15 attributes with a mean of 3 attribute values.

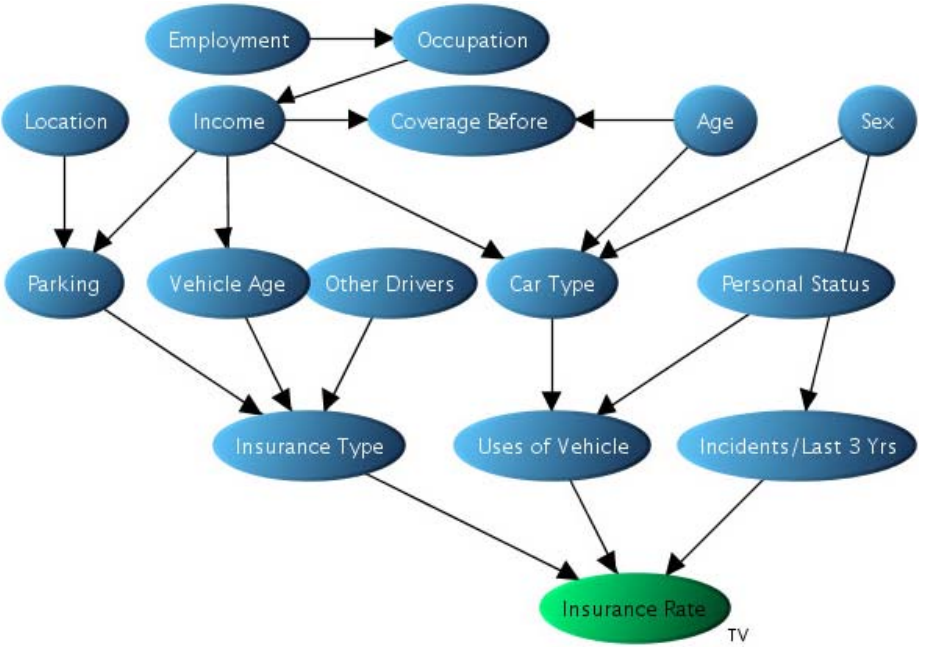


Fig. 1. Artificial vehicle insurance domain: Basic Bayesian network used for generating the synthetic evaluation data

The conditional probability tables of the network were initialized with uniform value distributions, i.e., each value in the table of an unconditioned node is equiprobable, and the conditional entries of a conditioned node are also defined by a uniform distribution.

In the experiments described below we compare three subgroup discovery algorithms: a standard beam search algorithm [2], the Apriori-SD method [12] and the proposed novel SD-Map algorithm. The beam search method and the Apriori-SD approach were implemented to the best of our knowledge based on the descriptions in the literature. All the experiments are performed on a 3.0 GHZ Pentium 4 PC machine running

Windows XP, providing a maximum of 1GB main memory for each experiment (the size of the main memory was never a limiting factor in the experiments).

4.1 Results

We generated data sets containing 1000, 10000 and 100000 instances. In each of these, there are exponentially numerous frequent attribute value combinations as the support threshold is reduced, considering the total search space of about 10^{10} patterns with respect to all combinations of the attribute values without internal disjunctions.

We measured the run-times of the algorithms by averaging five runs for each algorithm for each of the test data sets. For the beam search method we applied a quite low beam width ($w = 10$) in order to compare a fast beam search approach to the other methods. However, since beam search is a heuristic discovery method it could never discover all of the interesting subgroups compared to the exhaustive SD-Map and Apriori-SD algorithms in our experiments.

Since the minimal support threshold is the parameter that is common to all search methods we can utilize it in order to compare their scalability: In the experiments, we used two different low support thresholds, 0.01 and 0.05, to estimate the effect of increasing the pruning threshold, and to test the performance of the algorithms concerning quite small subgroups that are nevertheless considered as interesting in some domains, e.g., in the medical domain. We furthermore vary the number of instances, and the number of attributes provided to the subgroup discovery method in order to study the effect of a restricted search space compared to a larger one. Thus, by initially utilizing 5, then 10 and finally 15 attributes, an exponential increase in the search space for a fixed data set could be simulated. We thus performed 18 experiments for each algorithm in total, for each description language variant. The results are shown in the tables in Figure 2. Due to the limited space we only include the results for the 0.05 minimum support level with respect to the description language using selectors with internal disjunctions.

4.2 Discussion

The results of the experiments show that the SD-Map algorithm clearly outperforms the other methods. All methods perform linearly when the number of cases/instances is increased. However, the SD-Map method is at least faster by one magnitude compared to beam search, and about two magnitudes faster than Apriori-SD.

When the search space is increased exponentially all search method scale proportionally. In this case, the SD-Map method is still significantly faster than the beam search method, even if SD-Map needs to handle an exponential increase in the size of the search space. The SD-Map method also shows significantly better scalability than the Apriori-SD method for which the runtime grows exponentially for an exponential increase in the search space (number of attributes). In contrast, the run time of the SD-Map method grows in a much more conservative way. This is due to the fact that Apriori-SD applies the candidate-generation and test strategy depending on multiple scans of the data set and thus on the size of the data set. SD-Map benefits from its divide-and-conquer approach adapted from the FP-growth method by avoiding a large number of generate-and-test steps.

Runtime (sec.) - Conjunctive Description Language (No Internal Disjunctions)

MinSupport=0.01	1000 instances			10000 instances			100000 instances		
#Attributes	5	10	15	5	10	15	5	10	15
SD-Map	0.06	0.19	1.0	0.2	0.6	4.4	2.0	3.9	24.2
Beam search (w=10)	0.7	2.2	4.8	6.8	26.4	45.8	63.0	275.4	421.3
Apriori-SD	1.3	39.2	360.1	11.7	366.0	2336.1	108.7	3762.1	22128.5

Runtime (sec.) - Conjunctive Description Language (No Internal Disjunctions)

MinSupport=0.05	1000 instances			10000 instances			100000 instances		
#Attributes	5	10	15	5	10	15	5	10	15
SD-Map	0.02	0.05	0.14	0.2	0.4	1.1	2.0	3.7	9.2
Beam search (w=10)	0.5	1.8	3.1	4.6	12.2	25.7	49.4	173.0	295.8
Apriori-SD	0.3	2.3	6.0	2.4	23.2	63.7	23.7	229.5	632.6

Runtime (sec.) - Conjunctive Description Language (Internal Disjunctions)

MinSupport=0.05	1000 instances			10000 instances			100000 instances		
#Attributes	5	10	15	5	10	15	5	10	15
SD-Map (Negated S.)	0.05	0.4	8.5	0.3	1.4	38.7	2.8	6.1	144.5
SD-Map (Naive)	0.09	0.8	13.9	0.6	3.5	91.0	5.5	12.9	425.6
Beam search (w=10)	3.6	6.9	15.9	34.1	63.7	142.9	450.2	473.5	1093.6
Apriori-SD	6.9	124.3	7307.4	59.7	1000.2	23846.3	609.3	10148.3	208849

Fig. 2. Efficiency: Runtime of the algorithms for a conjunctive description language either using internal disjunctions or not

As shown in the tables in Figure 2 minimal support pruning has a significant effect on the exhaustive search methods. The run-time for SD-Map is reduced by more than half for larger search spaces, and for Apriori-SD the decrease is even more significant.

The results for the description language using internal disjunctions (shown in the third table of Figure 2) are similar to these for the non-disjunctive case. The SD-Map approach for handling internal disjunctions using negated selectors is shown in the first row, while the naive approach is shown in the second row of the table: It is easy to see that the first scales usually better than the latter. Furthermore, both outperform the beam search and Apriori-SD methods clearly.

5 Summary and Future Work

In this paper we have presented the novel SD-Map algorithm for fast but exhaustive subgroup discovery. SD-Map is based on the efficient FP-growth algorithm for mining association rules, adapted to the subgroup discovery setting. We described, how SD-Map can handle missing values, and how internal disjunctions in the subgroup description can be implemented efficiently. An experimental evaluation showed, that SD-Map provides huge performance gains compared to the Apriori-SD algorithm and even to the heuristic beam search approach.

In the future, we are planning to combine the SD-Map algorithm with sampling approaches and pruning techniques. Furthermore, integrated clustering methods for subgroup set selection are an interesting issue to consider for future work.

References

1. Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI Press (1996) 249–271
2. Klösgen, W.: 16.3: Subgroup Discovery. In: Handbook of Data Mining and Knowledge Discovery. Oxford University Press (2002)
3. Wrobel, S.: An Algorithm for Multi-Relational Discovery of Subgroups. In: Proc. 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-97), Berlin, Springer Verlag (1997) 78–87
4. Atzmueller, M., Puppe, F., Buscher, H.P.: Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery. In: Proc. 19th Intl. Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland (2005) 647–652
5. Lavrac, N., Kavsek, B., Flach, P., Todorovski, L.: Subgroup Discovery with CN2-SD. Journal of Machine Learning Research **5** (2004) 153–188
6. Friedman, J., Fisher, N.: Bump Hunting in High-Dimensional Data. Statistics and Computing **9**(2) (1999)
7. Scheffer, T., Wrobel, S.: Finding the Most Interesting Patterns in a Database Quickly by Using Sequential Sampling. Journal of Machine Learning Research **3** (2002) 833–862
8. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns Without Candidate Generation. In Chen, W., Naughton, J., Bernstein, P.A., eds.: 2000 ACM SIGMOD Intl. Conference on Management of Data, ACM Press (2000) 1–12
9. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proc. 20th Int. Conf. Very Large Data Bases, (VLDB), Morgan Kaufmann (1994) 487–499
10. Atzmueller, M., Puppe, F., Buscher, H.P.: Profiling Examiners using Intelligent Subgroup Mining. In: Proc. 10th Intl. Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2005), Aberdeen, Scotland (2005) 46–51
11. Kavsek, B., Lavrac, N., Todorovski, L.: ROC Analysis of Example Weighting in Subgroup Discovery. In: Proc. 1st Intl. Workshop on ROC Analysis in AI, Valencia, Spain (2004) 55–60
12. Kavsek, B., Lavrac, N., Jovanoski, V.: APRIORI-SD: Adapting Association Rule Learning to Subgroup Discovery. In: Proc. 5th Intl. Symposium on Intelligent Data Analysis, Springer Verlag (2003) 230–241
13. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. In: Proc. 4th Intl. Conference on Knowledge Discovery and Data Mining, New York, USA (1998) 80–86
14. Zimmermann, A., Raedt, L.D.: CorClass: Correlated Association Rule Mining for Classification. In Suzuki, E., Arikawa, S., eds.: Proc. 7th Intl. Conference on Discovery Science. Volume 3245 of Lecture Notes in Computer Science. (2004) 60–72
15. Jovanoski, V., Lavrac, N.: Classification Rule Learning with APRIORI-C. In: EPIA '01: Proc. 10th Portuguese Conference on Artificial Intelligence, London, UK, Springer-Verlag (2001) 44–51
16. Zimmermann, A., Raedt, L.D.: Cluster-Grouping: From Subgroup Discovery to Clustering. In: Proc. 15th European Conference on Machine Learning (ECML04). (2004) 575–577

Decision Trees for Hierarchical Multilabel Classification: A Case Study in Functional Genomics

Hendrik Blockeel[1], Leander Schietgat[1], Jan Struyf[1,2], Sašo Džeroski[3], and Amanda Clare[4]

[1] Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
{hendrik.blockeel, leander.schietgat, jan.struyf}@cs.kuleuven.be
[2] Dept. of Biostatistics and Medical Informatics, Univ. of Wisconsin, Madison, USA
[3] Department of Knowledge Technologies, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Saso.Dzeroski@ijs.si
[4] Department of Computer Science, University of Wales Aberystwyth, SY23 3DB, UK
afc@aber.ac.uk

Abstract. Hierarchical multilabel classification (HMC) is a variant of classification where instances may belong to multiple classes organized in a hierarchy. The task is relevant for several application domains. This paper presents an empirical study of decision tree approaches to HMC in the area of functional genomics. We compare learning a single HMC tree (which makes predictions for all classes together) to learning a set of regular classification trees (one for each class). Interestingly, on all 12 datasets we use, the HMC tree wins on all fronts: it is faster to learn and to apply, easier to interpret, and has similar or better predictive performance than the set of regular trees. It turns out that HMC tree learning is more robust to overfitting than regular tree learning.

1 Introduction

Classification refers to the task of learning from a set of classified instances a model that can predict the class of previously unseen instances. Hierarchical multilabel classification differs from normal classification in two ways: (1) a single example may belong to multiple classes simultaneously; and (2) the classes are organized in a hierarchy: an example that belongs to some class automatically belongs to all its superclasses.

Examples of this kind of problems are found in several domains, including text classification [1] and functional genomics [2]. In functional genomics, an important problem is predicting the functions of genes. Biologists have a set of possible functions that genes may have, and these functions are organized in a hierarchy. It is known that a single gene may have multiple functions.

Hierarchical multilabel classification (HMC) can be performed by just learning a binary classifier for each class separately, but this has several disadvantages.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 18–29, 2006.
© Springer-Verlag Berlin Heidelberg 2006

First, it is *less efficient*, because the learner has to be run $|C|$ times, with $|C|$ the number of classes, which can be hundreds or thousands. Second, it often results in learning from *strongly skewed class distributions*: among hundreds of classes, there are likely to be some that occur infrequently. Many learners have problems with strongly skewed class distributions [3]. Third, *hierarchical relationships* between classes are not taken into account. The constraint that an instance belonging to a class must belong to all its superclasses is not automatically imposed. Finally, from the knowledge discovery point of view, the learned models identify features relevant for one class, rather than identifying features with high *overall relevance*.

Some authors have therefore studied HMC as a separate learning task, and developed learners that learn a single model that predicts all the classes of an example at once (see below). These learners include a few decision tree approaches, but for these no in-depth empirical study has been presented up till now. In this paper we perform such an in-depth study, using datasets from functional genomics. Our study yields novel insights about the suitability of decision trees for HMC, and in particular gene function prediction.

In Section 2 we discuss previous work; in Section 3 we present the system used for the empirical study described in Section 4. In Section 5 we conclude.

2 Related Work

Much work in hierarchical multilabel classification (HMC) has been motivated by text classification. Rousu et al. [1] present the state of the art in this domain, which consists mostly of Bayesian and kernel-based classifiers.

Another application domain of HMC is functional genomics: a typical learning task is to learn a model that assigns to a gene a set of functions, selected from a hierarchy. Barutcuoglu et al. [2] recently presented a two-step approach where support vector machines are learned for each class separately, and then combined using a Bayesian learner so that the predictions are consistent with the hierarchical relationships; this solves one of the four issues mentioned above.

From the point of view of knowledge discovery, it is sometimes useful to obtain more interpretable models, such as decision trees, and that is the kind of approach we will study here.

Clare and King [4] presented a decision tree method for multilabel classification in the context of functional genomics. In their approach, a tree predicts not a single class but a vector of boolean class variables. They propose a simple adaptation of C4.5 to learn such trees: where C4.5 normally uses class entropy, their version uses the sum of entropies of the class variables. Clare [5] extended the method to predict classes on several levels of the hierarchy, assigning a larger cost to misclassifications higher up in the hierarchy, and presented an extensive evaluation on twelve datasets from functional genomics. We use this method as a reference to validate our own approach; we further refer to it as C4.5H.

Blockeel et al. [6] independently proposed a decision tree learner for HMC that is based on the concept of predictive clustering trees [7], where decision

trees are viewed as cluster hierarchies, and present preliminary experiments in text classification and functional genomics as a proof of concept. The approach has been used in some later work [8,9].

Until now these approaches have been evaluated mainly from the biologists' point of view, who commented on the discovered rules and their accuracy. No thorough performance evaluation from a machine learning point of view (what are the advantages over learning a single HMC tree over learning several regular trees?) has been made. Such an evaluation is in fact not trivial, when domain experts want to see as few rules as possible that predict as many classes as possible as correctly as possible. This work is the first thorough empirical comparison between HMC tree learning and learning multiple regular trees.

3 The Clus-HMC Approach

We first define the HMC task more formally; next, we describe the Clus-HMC system in detail.

3.1 Formal Task Description

We define the hierarchical multilabel classification task as follows:

Given: an instance space X and class hierarchy $(C, \leq_h)$, where C is a set of classes and $\leq_h$ is a partial order structured as a rooted tree, representing the superclass relationship (for all $c_1, c_2 \in C$: $c_1 \leq_h c_2$ if and only if c_1 is a superclass of c_2); a set T of examples (x_i, S_i) with $x_i \in X$ and $S_i \subseteq C$ such that $c \in S_i \Rightarrow \forall c' \leq_h c : c' \in S_i$; and some quality criterion q (which typically rewards models with high predictive accuracy and low complexity)
Find: a function $f : X \rightarrow 2^C$ (where 2^C is the power set of C) such that $c \in f(x) \Rightarrow \forall c' \leq_h c : c' \in f(x)$ and f maximizes q.

3.2 Clus-HMC: An HMC Decision Tree Learner

Fig. 1 presents the Clus-HMC algorithm. It is a variant of the standard greedy top-down algorithm for decision tree induction [10,11]. It takes as input a set of training instances T. The main loop of the algorithm searches for the best acceptable attribute-value test that can be put in a node. If such a test t^* can be found then the algorithm creates a new internal node labeled t^* and calls itself recursively to construct a subtree for each subset in the partition $\mathcal{P}^*$ induced by t^* on the training instances. If no acceptable test can be found, then the algorithm creates a leaf.

Up till here, the description is no different from that of a standard decision tree learner. However, decision tree learners normally predict only one target attribute, whereas an HMC tree needs to predict a set of classes. To achieve this, the following changes are made to the learning procedure [6].

First, the example labels are represented as vectors with boolean components; the i'th component of the vector is 1 if the example belongs to class c_i and 0

```
procedure Clus-HMC(T) returns tree
 1: (t , h , P ) = (none, ∞, ∅)
 2: for each possible test t
 3:     P = partition induced by t on T
 4:     h = Σ_{T_k}  (T_k/T) Var(T_k)
 5:     if (h < h ) ∧ acceptable(t, P)
 6:         (t , h , P ) = (t, h, P)
 7: if t ≠ none
 8:     for each T_k ∈ P
 9:         tree_k = Clus-HMC(T_k)
10:     return node(t , ⋃_k {tree_k})
11: else
12:     return leaf(v̄)
```

Fig. 1. The Clus-HMC induction algorithm

otherwise. It is easily checked that the arithmetic mean of a set of such vectors contains as i'th component the proportion of examples of the set belonging to class c_i. We define the variance of a set of examples as the average squared Euclidean distance between each example's label and the set's mean label.

The heuristic for choosing the best test in a node of the tree is then minimization of the average variance in the created subsets (weighted according to the size of the subsets, see line 4 of Fig. 1). This corresponds to the heuristic typically used when learning regression trees and to CART's Gini index [10], and is in line with the "predictive clustering trees" view [7]. The heuristic ensures that examples labelled with similar sets of classes tend to go into the same subset.

In the HMC context, it makes sense to consider similarity on higher levels of the hierarchy more important than similarity on lower levels. To that aim, we can use for the variance a weighted Euclidean distance $d(x_1, x_2) = \sqrt{\sum_k w_k \cdot (v_{1,k} - v_{2,k})^2}$, where $v_{i,k}$ is the k'th component of the class vector v_i of instance x_i, and the weights w_k decrease with the depth of the class c_k in the hierarchy (e.g., $w_k = w_0^{\text{depth}(c_k)}$). Consider for example the class hierarchy shown in Fig. 2, and two examples (x_1, S_1) and (x_2, S_2) with $S_1 = \{1, 2, 2/2\}$ and $S_2 = \{2\}$. Using a vector representation with consecutive components representing membership of class 1, 2, 2/1, 2/2 and 3, in that order, $d(x_1, x_2) = d_{\text{Euclidean}}([1, 1, 0, 1, 0], [0, 1, 0, 0, 0]) = \sqrt{w_0 + w_0^2}$.

A decision tree normally stores in a leaf the majority class for that leaf; this class will be the tree's prediction for examples arriving in the leaf. But in our case, since an example may have multiple classes, there is no "majority class". Instead, the mean $\bar{v}$ of the vectors of the examples in that leaf is stored; in other words, for each class c_i, the proportion of examples belonging to c_i is kept. An example arriving in the leaf will be predicted to belong to class c_i if the i-th component of $\bar{v}$ is above some threshold t_i. To ensure that the predictions fulfill the constraint that whenever a class is predicted its superclasses are also predicted, it suffices to choose $t_i \leq t_j$ whenever $c_i \leq_h c_j$.

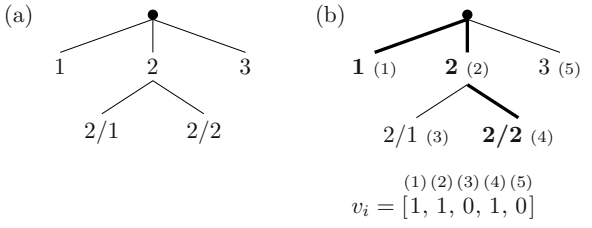


Fig. 2. (a) A toy hierarchy. Class label names reflect the position in the hierarchy, e.g., '2/1' is a subclass of '2'. (b) The set of classes {1,2,2/2}, indicated in bold in the hierarchy, and represented as a vector.

The predictive accuracy of the model is maximized by taking all $t_i = 0.5$, but as we are dealing with skewed class distributions, accuracy is not a very good evaluation criterion, and hence there is no good reason to try to maximize it. Precision-recall based evaluation is preferred in such cases [12]. Precision is the probability that a positive prediction is correct, and recall is the probability that a positive instance is predicted positive. When decreasing t_i from 1 to 0, an increasing number of instances is predicted as belonging to c_i, causing the recall for c_i to increase whereas precision may increase or decrease (with normally a tendency to decrease). Thus, a tree with specified threshold has a single precision and recall, and by varying the threshold for a single tree a precision-recall curve (PR curve) is obtained. Such curves allow us to evaluate the predictive performance of a model regardless of t.

Finally, the function *acceptable* in Fig. 1 verifies for a given test that the number of instances in each subset of the corresponding partition $\mathcal{P}$ is at least *mincases* (a parameter) and that the variance reduction is significant according to a statistical F-test.

The version of Clus-HMC described here is exactly the same as in Struyf et al. [8], except that the latter commits to a treshold of 0.5 whereas our version does not. Commitment to a fixed threshold causes models to be partially ordered in a precision-recall evaluation (e.g., a model may have higher recall but lower precision than another model), which is undesirable.

4 Experiments in Yeast Functional Genomics

The goals of our experiments are twofold. First, we wish to validate the "predictive clustering trees" approach to HMC, as implemented in Clus-HMC, by comparing its precision-recall behaviour to C4.5H, the state of the art in decision tree based HMC. Second, and most importantly, we evaluate the strengths and weaknesses of HMC tree learning as compared to learning a separate tree for each class. The expectation here is that HMC tree learning is faster and yields a tree that is more complex than an individual single-class tree, but less complex than the whole set of trees; one would further hope that this simplicity does not come at the cost of worse predictive performance.

```
1 METABOLISM
1/1 amino acid metabolism
1/2 nitrogen and sulfur metabolism
...
2 ENERGY
2/1 glycolysis and gluconeogenesis
...
```

Fig. 3. A small part of the hierarchical FunCat classification scheme

Table 1. Dataset properties: number of instances $|D|$, number of attributes $|A|$

Dataset	$\|D\|$	$\|A\|$	Dataset	$\|D\|$	$\|A\|$
D_1 Sequence (seq)	3932	478	D_7 DeRisi et al. (derisi)	3733	63
D_2 Phenotype (pheno)	1592	69	D_8 Eisen et al. (eisen)	2425	79
D_3 Secondary structure (struc)	3851	19628	D_9 Gasch et al. (gasch1)	3773	173
D_4 Homology search (hom)	3867	47034	D_{10} Gasch et al. (gasch2)	3788	52
D_5 Spellman et al. (cellcycle)	3766	77	D_{11} Chu et al. (spo)	3711	80
D_6 Roth et al. (church)	3764	27	D_{12} All microarray (expr)	3788	551

4.1 Datasets

Saccharomyces cerevisiae (baker's or brewer's yeast) is one of biology's classic model organisms, and has been the subject of intensive study for years. Its genes have annotations provided by the Munich Information Center for Protein Sequences (MIPS) under their FunCat scheme for classifying the functions of the products of genes. FunCat is a hierarchical system of functional classes. A small part of this hierarchy is shown in Fig. 3. Many yeast genes are annotated with more than one functional class.

We use the 12 datasets from [5]. An overview of these datasets is given in Table 1. The different datasets describe different aspects of the genes in the yeast genome. Five types of bioinformatic data for yeast are considered: sequence statistics (D_1), phenotype (D_2), predicted secondary structure (D_3), homology (D_4), and expression as measured with microarray chips (D_5 – D_{12}). The biologists' motivation for this is that different sources of data should highlight different aspects of gene function. More information on how the datasets were constructed and relevant references to the literature can be found in [5].[1]

Each gene in the datasets is annotated with one or more classes selected from the MIPS FunCat hierarchical classification scheme. The annotations and classification scheme available on 4/24/2002 were used. The hierarchy has 250 classes: 17 at the first level, 102 at the second, 89 at the third, and 42 at the fourth level.

4.2 Method

Clare [5] presents models trained on 2/3 of each dataset and tested on the remaining 1/3. In our experiments we use exactly the same training and test sets.

[1] Available together with the datasets at `http://www.aber.ac.uk/compsci/Research/bio/dss/yeastdata/`

To evaluate C4.5H, we computed the precision and recall of Clare's models. These models were presented as rules derived from the trees.

Clus-HMC results were obtained as follows. The weights used for the weighted Euclidean distance were chosen as $w_k = w_0^{\text{depth}(c_k)}$, with w_0 set to 0.75, and *mincases* was set to 5. The F-test stopping criterion takes a "significance level" parameter s, which was optimized as follows: for each out of 18 available values for s, Clus-HMC was run on 67% of the training set and its PR curve for the remaining 33% was constructed. The model having the largest area under this validation PR curve was then used to construct a PR curve for the test set. PR curves were constructed with non-linear interpolation between points [12].

To compare HMC tree learning to regular tree learning, we can use Clus on both sides. Indeed, the Clus system can be used for single classification just by reducing the class vector to a single component; its heuristic then reduces to CART's Gini index [10], and it performs comparably to regular decision tree learners. We refer to this version as Clus-SC. The results for Clus-SC were obtained in the same way as for Clus-HMC, but with a separate run for each class (including separate optimization of s for each class).

With 250 classes, each of which has its own PR curve, there is the question of how to evaluate the overall performance of a system. We can construct a single "average" PR curve for all classes together by counting instance-class-couples instead of instances. An instance-class couple is (predicted) positive if the instance has (is predicted to have) that class. The definition of precision and recall is then as before.

4.3 Results

Comparison with C4.5H. For each of the 12 datasets, average PR curves were generated and plotted against the points obtained by C4.5H. As we are comparing curves with points, we speak of a "win" for Clus-HMC when its curve is above C4.5H's point, a "loss" when it is below. Under the null hypothesis that both systems perform equally well, we expect as many wins as losses. For the average PR curves, we found 12 wins out of 12 for Clus-HMC. 4 representative plots are shown in Fig. 4. We have also included results for "Clus05", the predecessor of Clus-HMC where the s parameter was optimized for maximal precision given a fixed threshold of 0.5 [8]. Without the F-test optimization, the points of Clus05 would be on the Clus-HMC curve; small differences are due to the slightly different optimization criteria. It can clearly be seen that committing to a threshold of 0.5 kept Clus05 from achieving maximal precision.

We also made a class-by-class comparison: for each dataset and for each class for which C4.5H produced rules, we compared its PR to the Clus-HMC curve. Here we found 25 wins and 6 losses. Fig. 5 details the performance on the gasch1 dataset for the 7 classes predicted by C4.5H. Class 6/13/1 is the only class where Clus-HMC did not yield any classifiers strictly better than C4.5H. For Class 1 the C4.5H point is slightly above the Clus-HMC curve, yet Clus-HMC yields one classifier with the same precision but more than twice the recall.

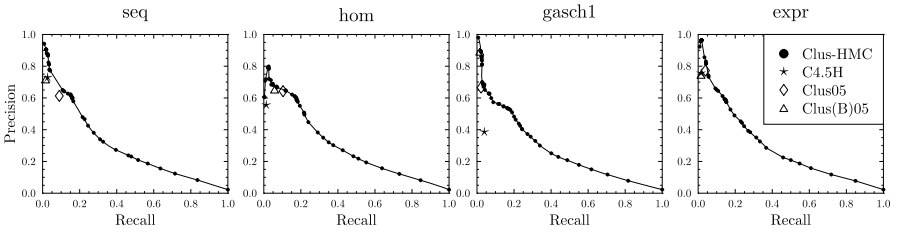


Fig. 4. Average precision/recall over all classes for Clus-HMC, C4.5H, and two versions of Clus-HMC's predecessor [8]

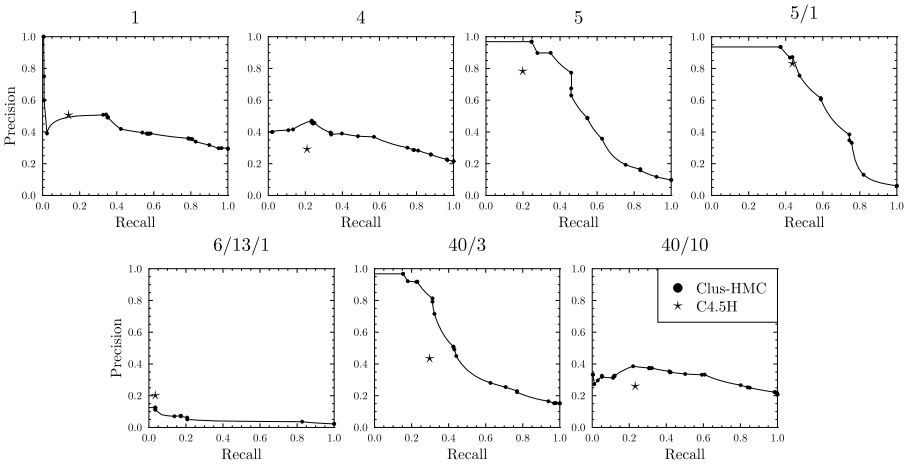


Fig. 5. Class by class comparison between Clus-HMC and C4.5H (gasch1)

Comparing the interpretability and precision/recall of individual rules (where a rule describes a single leaf of the tree), Clus-HMC also performs well. For instance, in the gasch1 dataset, for the class 40/3 (with prior frequency 14%), C4.5H learned two rules:

```
IF 29C_Plus1M_sorbitol_to_33C_Plus_1M_sorbitol___15_minutes <= 0.03 AND
  constant_0point32_mM_H2O2_20_min_redo <= 0.72 AND
  1point5_mM_diamide_60_min <= -0.17 AND
  steady_state_1M_sorbitol > -0.37 AND
  DBYmsn2_4__37degree_heat___20_min <= -0.67
THEN 40/3

IF Heat_Shock_10_minutes_hs_1 <= 1.82 AND
  Heat_Shock_030inutes__hs_2 <= -0.48 AND
  29C_Plus1M_sorbitol_to_33C_Plus_1M_sorbitol___5_minutes > -0.1
THEN 40/3
```

They have a precision/recall of 0.52/0.26 and 0.56/0.18, respectively. Clus-SC's most precise rule for 40/3, obtained by selecting a high threshold, is

```
IF Nitrogen_Depletion_8_h <= -2.74 AND Nitrogen_Depletion_2_h > -1.94 AND
   1point5_mM_diamide_5_min > -0.03 AND 1M_sorbitol___45_min_ > -0.36 AND
   37C_to_25C_shock___60_min > 1.28
THEN 40/3
```

with a precision/recall of 0.97/0.15. The second point on the PR curve turns out to represent the same rule with the last condition dropped; this rule scores 0.92/0.18. The best model consisting of two rules scores 0.92/0.23, and the best 3-rule model 0.81/0.31.

These results confirm that under precision-recall evaluation Clus-HMC performs at least as well as C4.5H, and can thus be considered a state-of-the-art HMC tree learner.

Comparison with Single Classification. We now turn to a comparison of HMC tree learning with learning separate trees for each class, using the Clus-HMC and Clus-SC instantiations. To limit the total runtime of Clus-SC, 40 classes were sampled from the hierarchy – 10 for each level.

For each dataset two average PR curves were generated: one for single classification, one for multilabel classification. Fig. 6 shows a few representative graphs. The single classification curve usually lies completely below the multilabel classification curve. Table 2 shows the difference in area under the PR curve (AUPRC) between Clus-HMC and Clus-SC; all differences are positive.

Fig. 7 shows some representative PR curves for gasch1. There are noticeable differences: sometimes Clus-SC performs better, sometimes Clus-HMC. So for individual classes the outcome is less clear-cut, but on average, Clus-HMC performs slightly better.

This is unexpected: one would think that Clus-SC has the advantage because it can learn a different optimal model for each class. Our original conjecture was that this was due to Clus-HMC performing better on the lower levels of the hierarchy.[2] But a per-level computation of the AUPRC difference (see again Table 2) does not confirm this: Clus-HMC tends to perform better overall, there is no clear correlation with depth in the hierarchy.

Further investigation revealed that Clus-SC tends to overfit more than Clus-HMC. Subtracting the area under the PR curve (AUPRC) obtained on the test set from that on the training set gives an indication of how strongly the approach overfits. Clus-SC scored a difference of 0.219, Clus-HMC 0.024.[3]

In hindsight, it makes sense that Clus-HMC overfits less than Clus-SC: overfitting 250 target values is simply more difficult than overfitting a single target

[2] Level four classes are very infrequent and therefore difficult to learn, but in Clus-HMC the parent classes may help in keeping the instances from class x/y/z together in the tree, and within a node with mainly x/y/z instances, a class x/y/z/u has a higher relative frequency.

[3] To make sure that this overfitting behaviour is not an artifact of our particular implementation, we also ran M5' from Weka [13] on the same datasets. M5' did not produce better results on the test set than Clus-SC and overfitted even more, with an AUPRC difference of 0.387. The tendency of M5' to overfit more than Clus-SC is consistent with our previous experience and with an earlier analysis by [14].

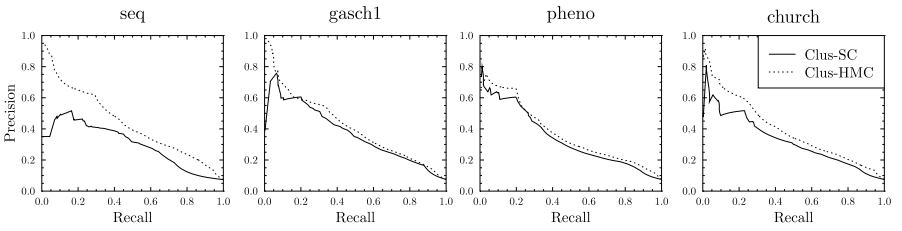


Fig. 6. Average precision/recall over all classes for Clus-SC and Clus-HMC

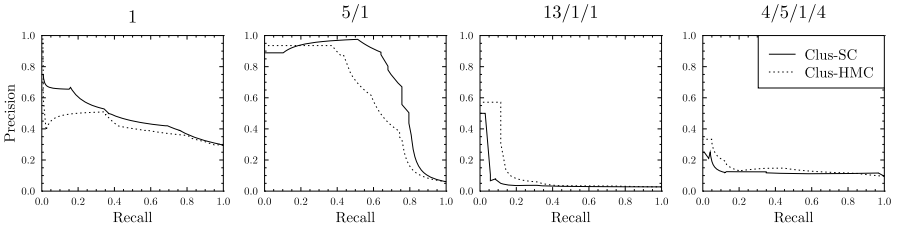


Fig. 7. Class by class comparison between Clus-SC and Clus-HMC (gasch1)

value. This is also visible in the tree sizes: Clus-HMC trees contain on average 24 nodes, whereas Clus-SC learns per dataset 250 trees with an average size of 33 nodes. In addition, Clus-HMC naturally takes dependencies between different classes into account (e.g., the constraints imposed by the hierarchy).

Clus-HMC runs slower than Clus-SC because it takes information about $|C|$ classes into acount, but on the other hand Clus-SC needs to be run $|C|$ times. On our twelve datasets Clus-HMC was 4.5 to 65 times faster than running Clus-SC for 250 classes, with an average speedup factor of 37.

Table 2. Overall and level-wise comparison of the area under the PR curve (AUPRC) of Clus-HMC and Clus-SC. Numbers are AUPRC(Clus-HMC) – AUPRC(Clus-SC).

Dataset	All	level 1	level 2	level 3	level 4
seq	0.142	0.135	0.086	0.056	0.025
pheno	0.030	0.025	0.001	0.022	0.010
struc	0.061	0.035	0.001	0.043	0.039
hom	0.057	0.058	0.032	0.028	0.036
cellcycle	0.038	0.037	-0.035	0.007	0.007
church	0.070	0.066	0.052	0.031	0.019
derisi	0.112	0.124	0.029	0.020	0.046
eisen	0.067	0.085	0.023	0.011	0.021
gasch1	0.044	0.027	0.041	0.032	0.018
gasch2	0.096	0.103	0.043	0.055	0.013
spo	0.095	0.102	0.022	-0.012	0.040
expr	0.099	0.071	0.062	0.031	0.021

5 Conclusions

We have conducted an empirical study of decision tree approaches to hierarchical multilabel classification.

Minor contributions of this study are, in the context of the predictive clustering trees (Clus) approach to HMC, (1) the description of a better tuned version of Clus-HMC (one that does not use thresholds that are suboptimal for precision-recall evaluation), and (2) a comparison showing that this version is at least as good as, and perhaps slightly better than, earlier HMC tree learning systems: it tends to yield rules with higher precision and recall and similar interpretability.

The major contribution however is the comparison between (a) learning a single tree that predicts all classes at once with a HMC-oriented algorithm, and (b) learning a separate decision tree for each class. We find that learning a single HMC tree is much faster than learning many regular trees, and it has the additional advantage of identifying features that are relevant for all the functions together (instead of separate features for each function). Obviously, a single HMC tree is also much more efficient to apply than 250 separate trees. Somewhat less expectedly, the HMC tree has on average a comparable predictive performance for each single class as a regular tree optimized for just that class. Our conjecture that the information contained in the hierarchy improves classification for infrequent classes in the lower parts of the hierarchy, which would partially explain this, was not confirmed. Instead, it turns out that the HMC approach is less susceptible to overfitting than the single classification approach. Fitting a model to many classes is indeed harder than fitting it to one class. Further, the HMC approach naturally takes into account dependencies between class membership, which may help it in making the right decisions during learning.

Given that HMC decision trees can yield better efficiency and interpretability without suffering a decline in predictive accuracy compared to learning separate trees, their use should definitely be considered in HMC tasks where interpretable models are desired.

Acknowledgements

H.B. and J.S. are post-doctoral fellows of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). L.S. is supported by a PhD grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). The authors thank Maurice Bruynooghe and Elisa Fromont for valuable suggestions and proofreading.

References

1. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Learning hierarchical multi-category text classification models. In De Raedt, L., Wrobel, S., eds.: Proceedings of the 22nd International Conference on Machine Learning, ACM Press (2005) 744 – 751
2. Barutcuoglu, Z., Schapire, R.E., Troyanskaya, O.G.: Hierarchical multi-label prediction of gene function. Bioinformatics **22**(7) (2006) 830–836

3. Weiss, G.M., Provost, F.J.: Learning when training data are costly: The effect of class distribution on tree induction. J. Artif. Intell. Res. (JAIR) **19** (2003) 315–354
4. Clare, A., King, R.: Knowledge discovery in multi-label phenotype data. In De Raedt, L., Siebes, A., eds.: 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2001). Volume 2168 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2001) 42–53
5. Clare, A.: Machine learning and data mining for yeast functional genomics. PhD thesis, University of Wales, Aberystwyth (2003)
6. Blockeel, H., Bruynooghe, M., Džeroski, S., Ramon, J., Struyf, J.: Hierarchical multi-classification. In: Proceedings of the ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining (MRDM 2002). (2002) 21–35
7. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Proceedings of the 15th International Conference on Machine Learning. (1998) 55–63
8. Struyf, J., Džeroski, S., Blockeel, H., Clare, A.: Hierarchical multi-classification with predictive clustering trees in functional genomics. In: Progress in Artificial Intelligence: 12th Portugese Conference on Artificial Intelligence. Volume 3808 of Lecture Notes in Computer Science., Springer (2005) 272–283
9. Struyf, J., Vens, C., Croonenborghs, T., Dzeroski, S., Blockeel, H.: Applying predictive clustering trees to the inductive logic programming 2005 challenge data. In: Inductive Logic Programming, 15th International Conference, ILP 2005, Late-Breaking Papers, Institut für Informatik der Technischen Universität München (2005) 111–116
10. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth, Belmont (1984)
11. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann (1993)
12. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. Technical report, University of Wisconsin, Madison (2005)
13. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)
14. Torgo, L.: A comparative study of reliable error estimators for pruning regression trees. In Coelho, H., ed.: Proceedings of the Iberoamerican Conference on AI (IBERAMIA-98), Springer (1998)

Clustering Scientific Literature Using Sparse Citation Graph Analysis

Levent Bolelli[1], Seyda Ertekin[1], and C. Lee Giles[1,2]

[1] Department of Computer Science and Engineering,
The Pennsylvania State University,
University Park, PA, 16802, USA
{bolelli, sertekin}@cse.psu.edu
[2] College of Information Sciences and Technology,
The Pennsylvania State University,
University Park, PA, 16802, USA
giles@ist.psu.edu

Abstract. It is well known that connectivity analysis of linked documents provides significant information about the structure of the document space for unsupervised learning tasks. However, the ability to identify distinct clusters of documents based on link graph analysis is proportional to the density of the graph and depends on the availability of the linking and/or linked documents in the collection. In this paper, we present an information theoretic approach towards measuring the significance of individual words based on the underlying link structure of the document collection. This enables us to generate a non-uniform weight distribution of the feature space which is used to augment the original corpus-based document similarities. The experimental results on the collection of scientific literature show that our method achieves better separation of distinct groups of documents, yielding improved clustering solutions.

1 Introduction

Document clustering refers to the task of extracting latent groupings in text databases. In broad terms, clustering is an optimization problem that attempts to find a partition of the document collection such that the items belonging to the same cluster are as similar as possible (cluster compactness) and the discovered clusters as separate as possible (cluster distinctness) based on a specified (dis)similarity metric within the high dimensional space that the document objects exist. In document collections where the only measure of similarity is textual content of documents, the traditional approach is the identification of meaningful features from documents and selection of the subset of features from the text corpus that yield better separation of distinct groups of documents. The clustering algorithm is then applied to this lower dimensional input space to discover distinct clusters.

The rapidly growing world wide web and the increasing volume of scientific literature available in digital format on the web has stimulated supervised and unsupervised data mining research to focus on linked documents. For linked documents, in addition to the textual content similarity, which can be thought of as an *implicit similarity*, we now have the link graph of the documents that depicts the *relatedness information*

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 30–41, 2006.
© Springer-Verlag Berlin Heidelberg 2006

conveyed by the authors of the digital content. Conventional clustering algorithms use attribute information to group documents under the assumption that two documents are related to each other if they have similar attribute values. However, relational data are richer in structure, hence provide more information available to disambiguate groupings. Therefore, link structure analysis has been studied extensively and has shown to be a significant aid for both supervised and unsupervised data analysis tasks. In this paper, we focus on clustering in the collection of scientific literature to discover topical groupings of papers using the textual content of papers combined with the citation graph of the collection. In a citation graph, papers are represented as vertices of the graph and citations as directed edges between citing and cited documents. The papers and citation graph have been obtained from CiteSeer's[1] repository.

CiteSeer [6] is a scientific literature digital library that has grown to index over 740.000 academic publications in Computer Science and related fields. Citations of the papers are extracted and linked to cited papers by Autonomous Citation Indexing [16]. The citation graph that is constructed through this process provides wealth of information since citations in research publications represent an important knowledge source regarding the context of scientific work. The citation relationships have been shown to be a valuable resource for a number of tasks such as ranking search results, identification of related research documents, trend analysis and social network analysis. Besides topical relevance, there have been identified multiple factors influencing citations, including the desire to publicize own research [10] and promoting own field, author's ability to access the document [15] and to read the language that it is written in [24]. Regardless of the reason for citations, comparatively, citation relationships between scientific documents convey a more valuable information than a collection of linked web documents. However, the citation graph itself can have limited clustering performance in digital libraries due to the following issues:

1) *Cited Document Availability*. CiteSeer collects the papers by crawling the web. Thus, the citations of a paper (i.e. target papers) may not be locally available in CiteSeer's repository due to several reasons: a) the citations may not be available on the web, b) they may just not have been crawled, or c) they may not be related to Computer Science or a similar field and may not be indexed. If any of these cases is true, the citations point to virtual metadata records that is identified by the extracted fields of the citation, including title, authors, publication venue, etc. However, the unavailability of the textual content of the cited papers prevents detailed analysis on the semantic similarity between the citing and cited papers.
2) *Identity Uncertainty*. Citations are references to unique documents, but their representations may vary, and finding the best matches for citations is a problem known as identity uncertainty [19]. The task of ACI is to uncover the identity of the paper that a citation refers to in order to group together similar citations to the same document, and to link citations to real documents – those that exist inside the ACI system and those that are yet to be crawled. Although ACI has been highly effective, it is still possible that distinct representations of the same citation may be mapped to different documents, or two citations to different papers be linked to the same target paper.

[1] http://citeseer.ist.psu.edu

The aforementioned reasons lead us to use only the citations where both the cited and citing documents are available in the collection, which sparsifies the link graph significantly. In this paper, we show that taking an information theoretic approach towards textual content analysis of pairs of documents with citation relationships provides a significant improvement in the discovery of document clusters. Further, we believe that the methodology presented here is applicable to web document collections where similar link constraints can be observed. One example is hierarchical clustering of documents where lower level taxonomies may not exhibit strong connectivity. Another application domain is search engine result clustering [9], an often employed technique to facilitate users' quick browsing through search results. Both applications suffer from the lack of sufficient links between the documents in a given subspace of the entire collection, which can be addressed by the algorithm proposed here.

2 Related Work

Document Clustering algorithms can be broadly categorized as text-based [20,2,21], link-based [22,11] and hybrid [23,18,14] approaches. In the domain of linked documents, link analysis for clustering and classification purposes has generally been studied in the context of web documents. PageRank [1] and HITS [13] are two of the most popular algorithms showing the importance of link structure for analyzing associations between documents.

For merging text-based and link-based information, [5] and [3] use generative probabilistic models of document content and connectivity. He et al. [23] use the hyperlink structure to cluster web pages using spectral graph partitioning. In their work, the link graph is used as the dominant source of similarity between documents, and the link-based similarity measures are augmented by textual content similarity and co-citation similarity. [14] propose a probabilistic model of link structure based on the cluster membership. The model is optimized based on observed data where the attributes determine the group membership and group membership determines the link structure. Modha et al. [17] propose an algorithm for clustering hypertext documents by using both the document contents and link structure. The algorithm uses an extended version of the classical Euclidean K-means clustering algorithm that performs clustering based on word similarity, in-link similarity and out-link similarity. The effect of each similarity is controlled by a parameter, which needs to be explicitly set by the user.

A number of algorithms have been proposed for *link prediction*, which is the task of identifying the missing entity or entities of a partially observed link by using the existing observation of the data sample available in the domain. [4] uses directed graphical models (Bayesian Networks and Probabilistic Relational Models) to represent a probabilistic model of both links and data object attributes. A comparison of various machine learning approaches for link prediction/completion is given in [8]. One major drawback of model-based link prediction is the dependence on the training data. That is, the learner's builds a probabilistic model on the training data, it will lack confidence in the probabilities of the entities that have not been included in the training set.

3 Problem Description

Documents on similar topics exhibit specific characteristics that separate them from non-relevant documents. Similar documents cite each other and they contain some level of textual similarity, measured by the amount of overlapping words/phrases. Some of those terms are very general and are not useful for clustering purposes. Some, on the other hand, are highly correlated with the topics of the papers and they are very valuable for identifying topical clusters in the collection. Although both textual content and link structure can be used independently for topical clustering, an algorithm that merges both sources of heterogeneous data has the potential of yielding better clustering solutions than using either data source alone. If the link structure of the documents are dense enough, then link based clustering, augmented by textual content, will generally yield well separated clusters. On the other hand, in situations where link graph is sparse, access to linking and/or linked documents is limited, or there is some sort of ambiguity in the link structure itself, the link graph can not be used as the dominant source of clustering. Thus, it is crucial to find a text-based clustering solution that incorporates information from the available link structure as well. Our work addresses this problem and provides an algorithm that bridges the disconnect between text and citations of papers by discovering the set of words that are most informative in terms of identifying citation relationships. We then place higher emphasis on such words in the clustering stage, and discover topical clusters in the citation-augmented feature space.

4 Algorithm

In this section, $d_i \in \mathbb{R}^n$ denotes the m documents in the collection and C denotes the non-symmetric citation matrix where $C_{ij} = 1$ if d_i cites d_j, and zero otherwise. Each document is represented as a vector in the feature space. Following L_2 normalization of the document vectors so that each $\|\boldsymbol{d_i}\| = 1$, we generate a similarity matrix S from the cosine similarities of each document pair:

$$S_{ij} = cos(d_i, d_j) = \frac{\boldsymbol{d_i^T} \cdot \boldsymbol{d_j}}{\|\boldsymbol{d_i}\| \cdot \|\boldsymbol{d_j}\|} \tag{1}$$

We then calculate, for each citing document, the average distance of its citations using the similarity matrix S and the citation graph as follows:

$$\mathbb{D}_i = \frac{\sum_{j=1}^{n} S_{ij} \cdot C_{ij}}{k_i} \tag{2}$$

where $\mathbb{D}_i$ represents the average citation distance(ACD) of document d_i, and k_i is the total number of citations of document d_i that is present in the collection. In this definition, only the citations in the collection can affect the distance metric, since, for missing citations, we do not have the text of the document and hence, S_{ij} will be zero. We are interested in evaluating the significance of each word by comparing its popularity both in document pairs *with* and *without* citations. To achieve this goal, the ACDs enable us to view the document space from these two perspectives by populating the following

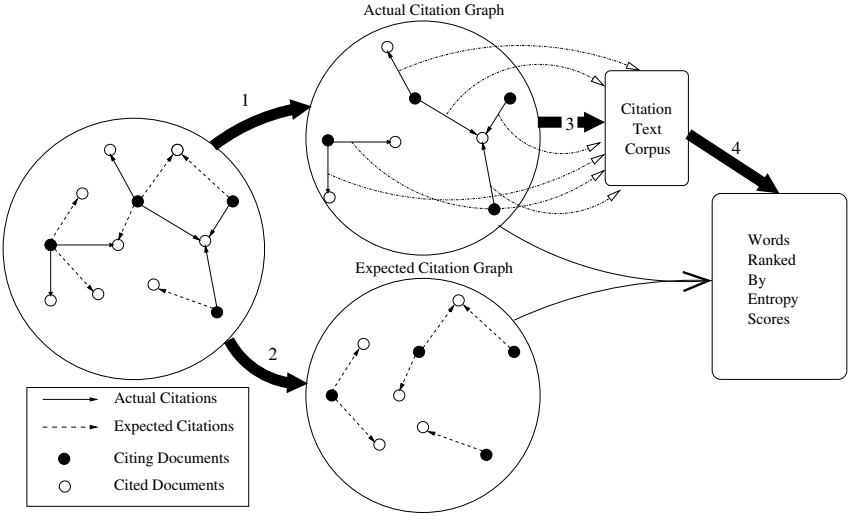
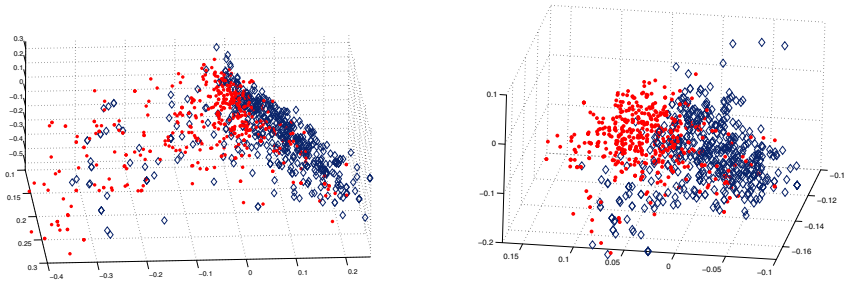


Fig. 1. Schematical view of the algorithm. The ACDs are used to find the expected citations and the given link structure is split into Actual Citation Graph G^A and Expected Citation Graph G^E (Steps 1 & 2). The set of words appearing in both in citing and cited documents in G^A are inserted in the Citation Text Corpus T (Step 3). For each word in T, we use the link and word co-occurence information from G^A and G^E to calculate the expected entropy loss scores (Step 4).

two sets: The first set, G^A is the *Actual Citation Graph* and is populated with the citing papers and their citations. This set is the collection of documents that form the citation graph. The second set, G^E, is the *Expected Citation Graph* and it is populated using the $\mathbb{D}_i$'s in the following fashion. For each document d_i having k_i citations (i.e. the citing documents in G^A), we select k_i documents that are **not** cited by d_i and is separated from d_i by a distance closest to a radius $\mathbb{D}_i$. That is, for each citing document d_i, we find k_i documents such that their content-wise similarity to d_i suggests that d_i should also be citing these documents, but no such citation exists in the graph for d_i. This set of documents is called *Expected Citation Graph* since we would expect these citations to exist based on the textual content of the papers.

Algorithm. Non-uniform Feature Weighting

1. Populate G^A with the documents in the citation graph
2. Initialize $G^E \leftarrow \emptyset$, $T \leftarrow \forall t_{ij}$ for $C_{ij} = 1$
3. **for** each citing document d_i in G^A with k_i citations **do**
4. $G^E \leftarrow G^E \cup \{k_i$ not-cited documents of d_i closest to $\mathbb{D}_i\}$
5. **end**
6. **for** each $t_p \in T$ **do**
7. E_p = Entropy loss calculated from equation 5.
8. $\bar{w}(d_i, t_p) \leftarrow (1 - \lambda) \cdot w(d_i, t_p) + \lambda \cdot E_p, \forall d_i \in \{d_1, d_2, \cdots, d_m\}$
9. **end**



(a) Original Document Space

(b) Document vectors adjusted by Entropy loss

Fig. 2. Effect of integrating entropy scores of citation corpus. Documents are mapped to 3D space by Singular Value Decomposition (SVD).

After populating G^E with the citing documents in G^A and their respective *expected citations*, the sets G^A and G^E have exactly the same number of edges, since we restrict G^E to contain the same linking vertices as G^A and insert exactly the same number of (expected) citations to it. This way, we enable each vertex (i.e. citing document) to be equally represented both in G^A and G^E. We then collect the common words between citing and cited documents in G^A. We do not consider the shared terms in the document pairs that are in G^E, since our aim is to identify the importance of the terms that appear in actual citation relationships.

We use expected entropy loss measure [7] to calculate the amount of information that each term in T conveys about citations. Our intuition is to find a numerical representation of the importance of each feature that is shared by the documents linked together. This also enables us to learn what makes document A cite document B and not cite C, although B and C may also be similar based on textual content. Clearly, it is not possible for a paper about, say, *data mining* to cite all the literature about this topic. Due to this fact, since lack of a citation can't be regarded as irrelevance, if we can identify the terms that influence citations, we can reflect the information obtained from the citations to better utilize the textual information of the documents for clustering purposes. If a word occurs frequently between citing and cited documents in G^A, but not in the expected citations in G^E, this word is regarded as a good candidate for being a topical word and is emphasized in the clustering algorithm. This approach serves as a means of eliminating one shortcoming of clustering algorithms; that is, each feature is weighted based on some corpus statistics and almost all clustering algorithms treat the attributes of data objects uniformly. We break this uniformity by reflecting the information obtained from the citation graph by scoring the shared terms of the citations using expected entropy loss.

4.1 Expected Entropy Loss

Given a text corpus comprising of n distinct features and k categories, expected entropy loss measures amount of categorical discriminative power of each feature in the dataset.

In formal definition, let C^A and C^E be the events of a sample being a member of the specified class, where the superscripts A and E refer to the actual and expected citation graphs, respectively. A sample in our case is the shared term between citing and cited documents. The prior entropy of the class distribution is

$$e = -P(C^A)lgP(C^A) - P(C^E)lgP(C^E) \tag{3}$$

The posterior entropy of the class distribution when feature f is present in the citation text corpus is

$$e_f = -P(C^A|f)lgP(C^A|f) - P(C^E|f)lgP(C^E|f) \tag{4}$$

The posterior entropy of the class distribution when feature f is absent in the corpus is denoted as $e_{\bar{f}}$ and can be found in a similar manner. Thus, the posterior expected entropy is $e_f P(f) + e_{\bar{f}} P(\bar{f})$ and expected entropy loss is defined as

$$Ent.Loss(f) = e - (e_f P(f) + e_{\bar{f}} P(\bar{f})) \tag{5}$$

which is always positive for every feature f.

4.2 Feature Weight Adjustment

The citation text corpus T contains the shared words between citing and cited documents in G^A (which is a subset of the original feature space) and we use this subset to realign the document vectors. Expected entropy loss based ranking of the most and least informative words in the corpus T is given in Table 1. It can be noted that more meaningful and topic bearing terms rank higher than less informative terms. Hence, by integrating the entropy loss information into the document vector representations, it is possible to achieve better separation of the distinct clusters. For each word in T, we update each document vector containing that feature as follows:

$$\bar{w}(d_i, f_j) \leftarrow (1 - \lambda) \cdot w(d_i, f_j) + \lambda \cdot Ent.Loss(f_j) \tag{6}$$

for $i = [1 \cdots n]$, $\forall f_j \in \boldsymbol{d_i}$. $w(d_i, f_j)$ represents the original Term Frequency-Inverse Document Frequency (TF-IDF) score of feature f_j in document d_i and λ is a parameter that adjusts the effect of the information gain of the feature on the final weight, which can also be thought of as relative bias of that term in the document. $\lambda = 0$ refers to the original weighting scheme and $\lambda = 1$ corresponds to purely entropy score based weighting. Hence, λ has the effect of proportionally reducing the significance of the features that don't exist in the citation corpus. Following the weight updates of the features of all documents, the document vectors are re-normalized to unit length. We then perform clustering on the updated document vectors. A visual representation of the effects of the weight readjustment is shown in Figure 2 for categories 1 and 4 of our dataset, which are the two clusters most difficult clusters to separate. It can be seen that comparably cleaner separation of the clusters can be achieved by entropy based weight readjustment of the features for these most overlapping categories. Computationally, given a dataset with N documents, C citations and a text corpus of T, the complexity of generating the similarity matrix and formation of the expected citation graph is $O(TN^2)$ and the calculation of the expected entropy losses is bounded by $O(CT)$. So the overall complexity of the algorithm is $O(T(N^2 + C))$.

Table 1. Features ranked by decreasing expected entropy loss

Rank	Feature
1.	automata
2.	radio
3.	collapse
4.	realtime
5.	switchboard
6.	tcp
7.	molecular
8.	fluctuate
9.	grayscale
10.	dendogram
...	...
...	...
...	...
5547.	statement
5548.	quinlan
5549.	roth

5 Experiments

We used a selection of 7227 papers from CiteSeer's repository as our dataset. The papers are split into 5 groups based on their publication venues. The categorical distribution of the publication venues is shown in Table 2. We selected the first 1000 words of each paper, resulting in a text corpus of 9601 distinct features after preprocessing the text by stemming, stop word and infrequent word removal. The clustering is performed both using the original TF-IDF scores of words and the scores augmented by the entropies of the words. A total of 4404 citation relationships exist between the papers in the dataset. The text corpus T of the citation relationships consists of distinct 5449 words. We used the Cluto [12] clustering toolkit in our experiments. Cluto implements some of the most widely used clustering algorithms in the literature, including agglomerative, divisive and graph-based techniques and hence, provides good baseline comparisons.

5.1 Evaluation Metric

The clustering performance is evaluated by comparing the predicted cluster of each document with the categorical labels (venues) from the document corpus. We used the standard F_1 and entropy measures as our evaluation criteria. F_1 measure combines precision (p) and recall (r) with equal weight in the form of $F_1(p, r) = \frac{2.p.r}{p+r}$. We report results both on Macro-averaged F_1 and Micro-averaged F_1 scores. The key difference between those two F_1 measures is that macro-averaging gives equal weight to each cluster, whereas micro-averaging equally weights each document. The cluster entropy measure shows the distribution of various classes of documents within each cluster. For each cluster C_i of size n_i, the entropy of this cluster is defined as

$$E(C_i) = -\frac{1}{log\ k} \sum_{j=1}^{k} \frac{n_i^j}{n_i} log \frac{n_i^j}{n_i} \tag{7}$$

Table 2. Dataset Venue Distribution

Cluster 1		Cluster 2		Cluster 3		Cluster 4		Cluster 5	
Venue	Samples	Venue	Samples	Venue	Samples	Venue	Samples	Venue	Samples
AAAI	662	POPL	599	ICCV	682	ICML	990	VLDB	1049
IJCAI	599	PLDI	664	CVPR	830	ECML	211		
ICTAI	232					ML	80		
						KDD	629		
total	1493	*total*	1263	*total*	1512	*total*	1910	*total*	1049

where k is the number of classes in the dataset and n_i^j is the number of documents of the i^{th} class that were assigned to the jth cluster.

The entropy of the entire clustering solution is the average of the cluster entropies adjusted by their respective sizes, given by $\sum_{i=1}^{k} \frac{n_i}{n} E(C_i)$. A smaller entropy score indicates better clustering solution over the entire dataset.

5.2 Results on Four Criterion Functions

We evaluated our algorithm using the following four different similarity criterion functions. Each criterion function represents the objective that we try to optimize for discovering clusters. The first criterion, I_{sim}, is an *internal* similarity metric that tries to maximize the similarity between each document and the centroid of its assigned cluster. The second criterion function, E_{sim}, is an *external* approach that tries to separate the documents of each cluster from the entire collection. The *hybrid* approach, H_{sim}, tries to find a clustering solution by optimizing the inter-cluster (I_{sim}) and intra-cluster (E_{sim}) similarity metrics simultaneously. The final criterion, G_{sim}, uses the similarity graph of the documents and tries to find the optimum cuts of the graph using MinMaxCut algorithm.

$$maximize\ I_{sim}(S) = \sum_{r=1}^{k} \sum_{d_i \in S_r} cos(d_i, C_r) \tag{8}$$

$$minimize\ E_{sim}(S) = \sum_{i=1}^{k} n_i \frac{\sum_{v \in S_i, u \in S} cos(v, u)}{\sqrt{\sum_{v,u \in S_i} cos(v, u)}} \tag{9}$$

$$maximize\ H_{sim}(S) = \frac{I_{sim}}{E_{sim}} \tag{10}$$

$$minimize\ G_{sim}(S) = \sum_{m=1}^{k} n_m^2 \frac{cut(S_m, S - S_m)}{\sum_{d_i, d_j \in S_m} cos(d_i, d_j)} \tag{11}$$

The results of the clustering solutions using the four criterion functions are given in Table 3 for $\lambda = 0$ and $\lambda = 0.15$. S and $\mathbb{S}$ refer to the original and updated document similarities, respectively.

In all four criterions, we were able to achieve better clustering solutions using the entropy-based weight adjustments of the features. The most benefit can be observed

Table 3. Results on four different clustering criterion functions

	Internal Similarity		External Similarity		Hybrid		Graph-based	
	$I_{sim}(S)$	$I_{sim}(\$)$	$E_{sim}(S)$	$E_{sim}(\$)$	$H_{sim}(S)$	$H_{sim}(\$)$	$G_{sim}(S)$	$G_{sim}(\$)$
$F_1(Micro)$	80.7%	**85.7%**	80.8%	**81.9%**	76.1%	**82.5%**	81.5%	**84.4%**
$F_1(Macro)$	81.5%	**86.7%**	81.4%	**82.3%**	76.8%	**83.2%**	81.8%	**84.8%**
Entropy	36.7%	**28.7%**	37.5%	**33.5%**	41.8%	**32.8%**	36.3%	**34.3%**

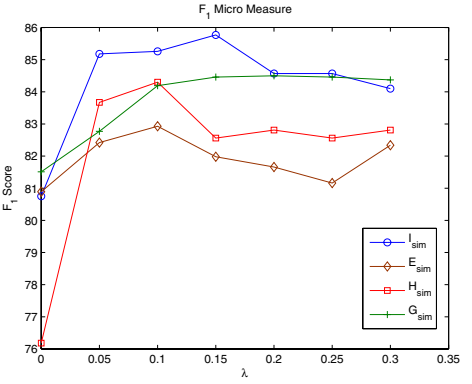


Fig. 3. F_1 Micro score variation based on λ

for I_{sim} and H_{sim} similarity metrics, indicating that similar documents are grouped into much compact clusters. This behavior is expected since the citations we used were mostly to the papers that are in the same category, hence we boosted the weights of the terms that collectively define their respective categories, hence maximizing the internal similarity of the documents of the same cluster. In Figures 3 and 4, we show the effect of varying λ on F_1 and entropy scores of the clustering solution for all criterion functions. Even for the $\lambda = 0.05$ case which indicates only a slight support from the

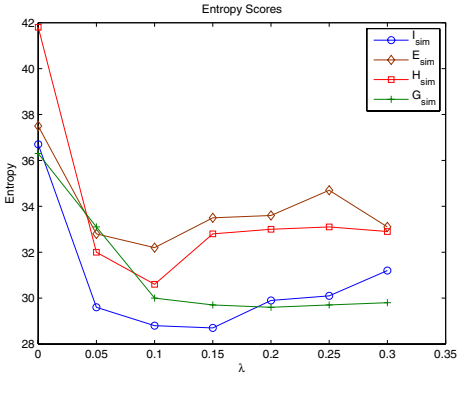


Fig. 4. Entropy score variation based on λ

entropies on the feature values, all four criterion functions achieve significant accuracy improvement. Further increasing λ over 0.25 either has no, or negative effect on the clustering solution.

Since the entropy values are needed for the *bias* effect on feature weights, increasing λ beyond a certain point starts to cause a dominating effect on the document vectors. In that case, the documents containing just a couple of common words (i.e. "database", "collection", "learning") tend to group together, causing an adverse effect. It is therefore desirable to keep λ at values that is sufficient enough to contribute to the weights without significantly modifying them.

6 Conclusions

Most clustering algorithms assume that the components of data objects are independent and identically distributed. This assumption has led to the design of numerous supervised and unsupervised learning algorithms to work on such "flat" data, where each data instance is a fixed length vector of attribute values. For data sets where the data set has richer structure, such as hyperlinks in web documents and citations in scientific literature, an efficient and effective solution to incorporate the connectivity information in the clustering solution yields better clustering performance. In this paper, we presented an algorithm that incorporates the citation graph of a collection of scientific literature to the clustering solution to better identify distinct groups of documents. The existence and non-existence of citation relationships of papers are used to identify the most important topic-bearing words in the papers, based on expected entropy loss measure. We have shown that a feature weighting scheme incorporating the citation-based extraction of topically significant words and applying partial bias for those terms can effectively discover clusters of similar papers.

References

1. S. Brin and L. Page. The anatomy of a large scale hypertextual web search engine. In *7th WWW Conference*, 1998.
2. T. Chiu, D. Fang, J. Chen, Y. Wang, and C. Jeris. A robust and scalable clustering algorithm for mixed type attributes in large database environment. In *KDD '01*, pages 263–268, 2001.
3. David Cohn and Thomas Hoffmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *NIPS*, 2001.
4. L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational data. In *ICML'02*, pages 170–177, 2001.
5. Lise Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, pages 679–708, 2002.
6. C. Lee Giles, Kurt Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In *The 3rd ACM Conf. on Digital Libraries*, pages 89–98, 1998.
7. E. Glover, S. Lawrence G. Flake, A. Kruger, D. Pennock, W. P. Birmingham, and C. L. Giles. Improving category specific web search by learning query modifications. In *SAINT '01*, page 23, 2001.
8. Anna Goldenberg, Jeremy Kubica, Paul Komarek, Andrew Moore, and Jeff Schneider. A comparison of statistical and machine learning algorithms on the task of link completion. In *KDD Workshop on Link Analysis for Detecting Complex Behavior*, August 2003.

9. Z. Chen H-J. Zeng, Q-C. He, W-Y Ma, and J. Ma. Learning to cluster web search results. In *SIGIR'04*, pages 210–217, 2004.
10. K. Hayland. Self-citation and self-reference: credibility and promotion in academic publication. *Journal of the Academic Society for Information Science*, 54(3):251–259, 2003.
11. J. Hou and Y. Zhang. Utilizing hyperlink transitivity to improve web page clustering. In *Proc. of 14th Australasian database conference on Database technologies*, pages 49–57, 2003.
12. George Karypis. Cluto, 2002. http://glaros.dtc.umn.edu/gkhome/views/cluto/.
13. Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
14. J. Kubica, A. Moore, J. Schneider, and Y. Yang. Stochastic link and group detection. In *18th National Conference on Artificial Intelligence (AAAI'02)*, 2002.
15. Steve Lawrence. Online or invisible. *Nature*, 411(6837):521, 2001.
16. Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.
17. D. Modha and W. Spangler. Clustering hypertext with applications to web searching. In *11th ACM Conf. on Hypertext and Hypermedia*, 2000.
18. J. M. Neville and D. Jensen. Clustering relational data using attribute and link information. In *Text Mining and Link Analysis Workshop, 18th Int'l Conf. on Artificial Intelligence*, 2003.
19. H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty in citation matching. In *Advances in Neural Information Processing*, 2003.
20. R. Rastogi S. Guha and K. Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD '98*, pages 73–84, 1998.
21. Y. Gong W. Xu, X. Liu. Document clustering based on non-negative matrix factorization. In *SIGIR'03*, pages 267–273, 2003.
22. Yitong Wang and M. Kitsuregawa. Use link-based clustering to improve web search results. In *Second Int'l. Conf. on Web Information Systems Engineering*, December 2001.
23. C. Ding X. He, H. Zha and H. Simon. Web document clustering using hyperlink structures. *Computational Statistics and Data Analysis*, 41:19–45, 2002.
24. M. Yitzhaki. The language preference in sociology: measurements of 'language self-citation', 'relative own language preference indicator' and 'mutual use of languages'. *Scientometrics*, 41:243–254, 1998.

VOGUE: A Novel Variable Order-Gap State Machine for Modeling Sequences

Bouchra Bouqata, Christopher D. Carothers,
Boleslaw K. Szymanski, and Mohammed J. Zaki

CS Department, Rensselaer Polytechnic Institute, Troy, NY, USA
{bouqab, chrisc, szymansk, zaki}@cs.rpi.edu

Abstract. We present VOGUE, a new state machine that combines two separate techniques for modeling long range dependencies in sequential data: data mining and data modeling. VOGUE relies on a novel Variable-Gap Sequence mining method (VGS), to mine frequent patterns with different lengths and gaps between elements. It then uses these mined sequences to build the state machine. We applied VOGUE to the task of protein sequence classification on real data from the PROSITE protein families. We show that VOGUE yields significantly better scores than higher-order Hidden Markov Models. Moreover, we show that VOGUE's classification sensitivity outperforms that of HMMER, a state-of-the-art method for protein classification.

1 Introduction

Many real world applications, such as in bioinformatics, web accesses, and text mining, encompass sequential/temporal data with long and short range dependencies. Techniques for analyzing such types of data can be classified in two broad categories: sequence pattern mining [12], and data modeling via Hidden Markov Models (HMMs) [4, 8]. HMMs depend on the Markovian property, i.e., the current state i in the sequence depends only on the previous state j, which makes them unsuitable for problems where general patterns may display longer range dependencies. For such problems, higher-order and variable-order HMMs [8–10] have been proposed, where the order denotes the number of previous states that the current state depends upon. However higher-order HMMs often suffer from a number of difficulties, namely, *high state-space complexity*, *reduced coverage*, and sometimes even *low prediction accuracy* [3].

In this paper we present a new approach to temporal/sequential data analysis via a novel state machine, **VOGUE** (**V**ariable **O**rder **G**aps for **U**nstructured **E**lements). The first step of our method uses a new sequence mining algorithm, called **V**ariable-**G**ap **S**equence miner (VGS), to mine variable-length frequent patterns, that may contain different gaps between the elements. The second step of our technique uses the mined variable-gap sequences to build the VOGUE state machine. In fact, VOGUE models multiple higher order HMMs via a single variable-order state machine. Although VOGUE has a much wider applicability,

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 42–54, 2006.
© Springer-Verlag Berlin Heidelberg 2006

in this paper we apply it to a problem in biological sequence analysis, namely, multi-class protein classification. Given a database of protein sequences, the goal is to build a statistical model so that we can determine whether a query protein belongs to a given family (class) or not. Statistical models for proteins, such as profiles, position-specific scoring matrices, and hidden Markov models [4] have been developed to find homologs. However, in most biological sequences, interesting patterns repeat (either within the same sequence or across sequences) and may be separated by variable length gaps. Therefore a method like VOGUE that specifically takes these kind of patterns into consideration can be very effective. We show experimentally that VOGUE's modeling power is superior to higher-order HMMs while reducing the latter's state-space complexity, and improving their prediction capabilities. VOGUE also outperforms HMMER [4], a HMM model especially designed for protein sequences.

2 Related Work

HMMs have been proposed to model longer range dependencies. However, such models suffer from high state-space complexity, since a k-th order HMM, with alphabet Σ, can potentially have $|\Sigma|^k$ states. Estimating the joint probabilities of each k-th order state is also difficult. The all-k-order Markov model was proposed in [8], where one has to maintain a Markov model of order j (where the current state depends on the j previous states) **for all** $1 \leq j \leq k$. Three *post-pruning* techniques were proposed in [3] to improve the prediction accuracy and coverage, and to lower the state complexity of the all k-order Markov model. However, multiple models still have to be maintained.

In [9], mixed order Markov models were proposed. However, they rely on Expectation Maximization (EM) algorithms that are prone to local optima. Furthermore, mixed order Markov models depend on a mixture of bigrams over k consecutive previous states, whereas VOGUE automatically ignores irrelevant states. Another approach combines the mining of sequences with a Markov predictor for web prefetching [7], but it is tuned specifically for web usage mining since it relies on the knowledge of the site structure. In [10], a suffix tree is incorporated in the training of the HMM which is done by an EM algorithm. Although this algorithm reduces the state space complexity of an all-k-order HMM, it still uses the previous k states and relies on an EM method. The Episode Generating HMM (EGH) [6] is especially relevant. However, there are notable differences in the EGH approach versus VOGUE. Instead of building EGHs per subsequence, VOGUE is a *single* variable-order state machine incorporating all the frequent sequences. In VOGUE, the gap states have the notion of duration which enables our system to account for long range dependencies. The Hierarchical HMM (HHMM) approach in [2] extracts episodes (using sequence alignment methods) from which (left-to-right) HMMs are built. HHMM also emits a random or "any" symbol in a gap state. In contrast, VOGUE simultaneously models all non-consecutive patterns, as well as gap symbol and duration statistics.

3 VOGUE State Machine

As noted earlier, building higher order HMMs is not easy, since we have to estimate the joint probabilities of the previous k states in a *k-order* HMM. Also, not all of the previous k states may be predictive of the current state. Moreover, the training process is extremely expensive and suffers from local optima due to the use of an EM (also known as Baum-Welch) algorithm for training the model. VOGUE addresses these limitations. It first uses the VGS algorithm to mine variable-gap frequent sequences that can have g other symbols between any two elements; g varies from 0 to a maximum gap ($MAXGAP$). These sequences are then used as the estimates of the joint probabilities for the states used to seed the model.

Consider a simple example to illustrate our main idea. Let the alphabet be $\Sigma = \{A, \cdots, K\}$ and the sequence be $S = \mathbf{ABA}C\mathbf{B}D\mathbf{A}EF\mathbf{B}GH\mathbf{A}IJK\mathbf{B}$. We can observe that $A \rightarrow B$ is a pattern that repeats frequently (4 times), but with variable length gaps in-between. $B \rightarrow A$ is also frequent (3 times), again with gaps of variable lengths. A first-order HMM will fail to capture any patterns since no symbol depends purely on the previous symbol. We could try higher order HMMs, but they will model many irrelevant parts of the input sequence. More importantly, *no fixed-order HMM for* $k \geq 1$ *can model this sequence*, since none of them detects the variable repeating pattern between A and B (or vice versa). This is easy to see, since for any fixed sliding window of size k, no k-letter word (or k-gram) ever repeats! In contrast our VGS mining algorithm is able to extract both $A \rightarrow B$, and $B \rightarrow A$ as frequent subsequences, and it will also record how many times a given gap length is seen, as well as the frequency of the symbols seen in those gaps. This knowledge of gaps plays a crucial role in VOGUE, and distinguishes it from all previous approaches which either do not consider gaps or allow only fixed gaps. VOGUE models gaps via *gap states* between elements of a sequence. The gap state has a notion of state duration which is executed according to the distribution of length of the gaps and the intervening symbols.

The training and testing of VOGUE consists of three main steps: 1) **Pattern Mining** via the novel Variable-Gap Sequence (VGS) mining algorithm. 2) **Data Modeling** via our novel Variable-Order state machine. 3) **Interpretation** of new data via a modified Viterbi method, called VG-Viterbi, to model the most probable path through a VOGUE model. Details of these steps appear below.

3.1 Mining: Variable-Gap Sequences (VGS)

VGS is based on cSPADE [11, 12], a method for constrained sequence mining. Whereas cSPADE essentially ignores the length of and symbol distributions in gaps, VGS is specially designed to extract such patterns within one or more sequences. Note that whereas other methods can also mine gapped sequences [1, 11], the key difference is that *during mining* VGS explicitly keeps track of all the intermediate symbols, their frequency, and the gap frequency distributions, which are used to build VOGUE.

VGS takes as input the maximum gap allowed ($maxgap$), the maximum sequence length (k), and the minimum frequency threshold ($minsup$). VGS mines all sequences having up to k elements, with a gap of at most $maxgap$ length between any two elements, such that the sequence occurs at least $minsup$ times in the data. For example, let $S = ACBDAHCBADFGAIEB$ be an input sequence over the alphabet $\Sigma = \{A, \cdots, I\}$, and let $maxgap = 2$, $minsup = 2$ and $k = 2$. VGS first mines the frequent subsequences of length 1, as shown in Table 1. Those symbols that are frequent are extended to consider sequences of length 2, as shown in Table 2. For example, $A \rightarrow B$ is a frequent sequence with frequency $freq = 3$, since it occurs once with gap of length 1 (ACB) and twice with a gap of length 2 ($AHCB$ and $AIEB$). Thus the gap length distribution of $A \rightarrow B$ is $0, 1, 2$ as shown under columns $g = 0$, $g = 1$, and $g = 2$, respectively. VGS also records the symbol distribution in the gaps for each frequent sequence. For $A \rightarrow B$, VGS will record gap symbol frequencies as $C(2), E(1), H(1), I(1)$, based on the three occurrences. Since $k = 2$, VGS would stop after mining sequences of length 2. Otherwise, VGS would continue mining sequences of length $k \geq 3$, until all sequences with k elements have been mined.

Table 1. VGS: Subsequences of Length 1

	A	B	C	D	E	F	G	H	I
frequency	4	3	2	2	1	1	1	1	1

Table 2. VGS: Subsequences of Length 2

subsequence	*freq*	$g = 0$	$g = 1$	$g = 2$
$A \rightarrow C$	2	1	1	0
$A \rightarrow B$	3	0	1	2
$A \rightarrow D$	2	1	0	1
$C \rightarrow B$	2	2	0	0
$C \rightarrow D$	2	0	1	1
$C \rightarrow A$	2	0	1	1
$B \rightarrow D$	2	1	1	0
$B \rightarrow A$	2	1	1	0
$D \rightarrow A$	2	1	0	1

3.2 Modeling: Variable-Order State Machine

VOGUE uses the mined sequences to build a variable order/gap state machine. The main idea here is to model each *non-gap* symbol in the mined sequences as a state that emits only that symbol and to add intermediate gap states between any two non-gap states. The gap states will capture the distribution of the gap symbols and length. Let F be the set of frequent sequences mined by VGS, and let k be the maximum length of any sequence. While VOGUE can be generalized to use any value of $k \geq 2$, for clarity of exposition and lack of space we will illustrate the working of VOGUE using mined sequences of length $k = 2$. Let F_1 and F_2 be the sets of all frequent sequences of length 1 and 2, respectively, so that $F = F_1 \cup F_2$. Thus, each mined sequence $s_i \in F_2$ is of the form $s_i : v_f \rightarrow v_s$, where $v_f, v_s \in \Sigma$. Let $\Gamma = \{v_f | v_f \rightarrow v_s \in F_2\}$ be the set of all the distinct symbols in the first position, and $\Theta = \{v_s | v_f \rightarrow v_s \in F_2\}$ be the set of all the distinct symbols in the second position, across all the mined sequences $s_i \in F_2$. The VOGUE model is specified by the 6-tuple $\lambda = \{Q, \Sigma, A, B, \rho, \pi\}$ where each component is defined below.

Alphabet (Σ): The alphabet for VOGUE is $\Sigma = \{v_1, \cdots v_M\}$, where $|\Sigma| = M$ is the number of observations emitted over all states. The alphabet's size is

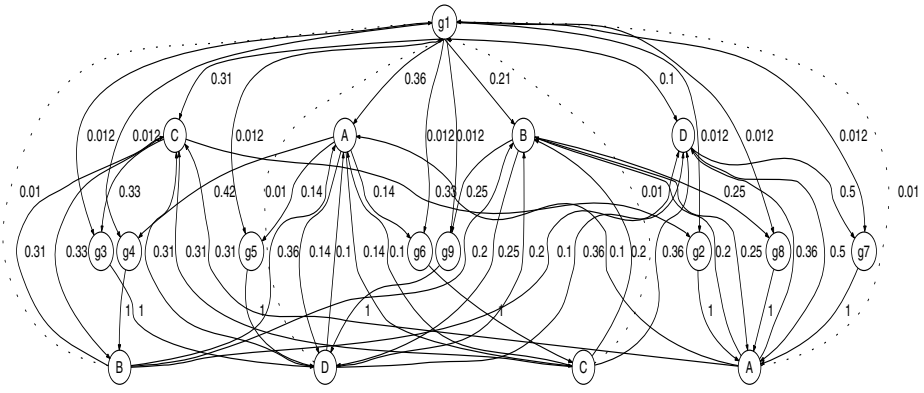


Fig. 1. VOGUE State Machine for Running Example

defined by the number of symbols that occur at least once in the training data, obtained as a result of the first iteration of VGS, as shown in Table 1. For our example S in Section 3.1, we have nine distinct frequent symbols, thus $M = 9$.

Set of States (Q): The set of states in VOGUE is given as $Q = \{q_1, \cdots, q_N\}$, where $|Q| = N = N_f + G_i + N_s + G_u$. Here, $N_f = |\Gamma|$ and $N_s = |\Theta|$ are the number of distinct symbols in the first and second positions, respectively. Each frequent sequence $s_i \in F_2$ having a gap $g \geq 1$ requires a gap state to models the gaps. G_i thus gives the number of gap states required. Finally $G_u = 1$ corresponds to an extra gap state, called *universal gap*, that acts as the default state when no other state satisfies an input sequence. For convenience let $Q = Q_f \cup Q_i \cup Q_s \cup Q_u$ be the partition of Q where the first N_f states belong to Q_f, the next G_i states belong to Q_i, and so on. For our example S in Section 3.1, we have $N_f = 4$, since there are four distinct starting symbols in Table 2 (namely, A, B, C, D). We also have four ending symbols, giving $N_s = 4$. The number of gap states is the number of sequences of length 2 with at least one occurrence with gap $g \geq 1$. Thus $G_i = 8$, $C \rightarrow B$ is the only sequence that has all consecutive ($g = 0$) occurrences. With one universal gap state $G_u = 1$, our model yields $N = 4 + 8 + 4 + 1 = 17$ states.

Transition Probability Matrix (A): The transition probability matrix between the states, $A = \{a(q_i, q_j) | 1 \leq i, j \leq N\}$, where $a(q_i, q_j) = P(q^{t+1} = q_j | q^t = q_i)$, gives the probability of moving from state q_i to q_j (where t is the current position in the sequence). The probabilities depend on the types of states involved in the transitions. The basic intuition is to allow transitions from the first symbol states to either the gap states or the second symbol states. The second symbol states can go back to either the first symbol states or to the universal gap state. Finally the universal gap state can go to any of the starting states or the intermediate gap states. We discuss these cases below.

Transitions from First States: Any first symbol state $q_i \in Q_f$ may transition to either a second symbol state $q_j \in Q_s$ (modeling a gap of $g = 0$) or to a gap state $q_j \in Q_i$ (modeling a gap of $g \in [1, maxgap]$). Let $s_{iy} : v_i \rightarrow v_y \in F_2$ be

a subsequence mined by VGS. Let $freq_i^g(y)$ denote the frequency of s_{iy} for a given gap value g, and let $freq_i(y)$ denote the total frequency of the sequence, i.e., $freq_i(y) = \sum_{g=0}^{maxgap} freq_i^g(y)$. Let $R = \frac{freq_i^0(j)}{\sum_{y \in Q_s} freq_i(y)}$ denote the fraction of gap-less transitions from q_i to q_j over all the transitions from q_i to $q_y \in Q_s$. The transition probabilities from $q_i \in Q_f$ are given as:

$$a(q_i, q_j) = \begin{cases} R & \text{for } q_j \in Q_s \\ \frac{freq_i(j)}{\sum_{y \in Q_s} freq_i(y)} - R & \text{for } q_j \in Q_i \\ 0 & \text{for } q_j \in Q_f \cup Q_u \end{cases}$$

Transitions from Gap States: Any gap state $q_i \in Q_i$ may only transition to second symbol state $q_j \in Q_s$. For $q_i \in Q_i$ we have:

$$a(q_i, q_j) = \begin{cases} 1 & \text{for } q_j \in Q_s \\ 0 & \text{for } q_j \in Q \backslash Q_s \end{cases}$$

Transitions from Second States: A second symbol state $q_i \in Q_s$ may transition to either first symbol state $q_j \in Q_f$ (modeling a gap of $g = 0$), or to the universal gap state $q_j \in Q_u$ (modeling other gaps). Let $T = \sum_{s_x \in F_2} freq(s_x)$ be the sum of frequencies of all the sequences in F_2. For $q_i \in Q_s$ we have:

$$a(q_i, q_j) = \begin{cases} 0.99 \times \frac{\sum_{q_y \in Q_f} freq_j(y)}{T} & \text{for } q_j \in Q_f \\ 0.01 & \text{for } q_j \in Q_u \\ 0 & \text{for } q_j \in Q_i \cup Q_s \end{cases}$$

Note that the transitions to universal gap have a small probability (0.01). Transitions back to first states are independent of q_i, i.e., the same for all $q_i \in Q_s$. In fact, these transitions are the same as the initialization probabilities described below. They allow the model to loop back after modeling a frequent sequence. Note that the values 0.99 and 0.01 above were chosen to allow (via pseudo-counts) for unseen symbols.

Transitions from Universal Gap: The universal gap state can only transition to the first states or the intermediate gap states. For $q_i \in Q_u$ we have:

$$a(q_i, q_j) = \begin{cases} 0.9 \times \frac{\sum_{q_y \in Q_f} freq_j(y)}{T} & \text{for } q_j \in Q_f \\ 0.1 \times \frac{1}{G_i} & \text{for } q_j \in Q_i \\ 0 & \text{for } q_j \in Q \backslash Q_f \end{cases}$$

Since the first states can emit only one symbol, we allow transitions from universal gap to intermediate gap states, to allow for other symbol emissions. This probability is at most 10% (empirically chosen) across all the gap states. In the remaining 90% cases, the universal gap transitions to a first state with probabilities proportional to its frequency.

Figure 1 shows transitions between states and their probabilities in VOGUE for our running example. Each gap state's duration is considered explicitly within a state. The notation g_i (e.g., g_3) is the name of the gap state between the elements of the sequence, (e.g., $C \rightarrow D$), and not the value of the gap. The symbol states, on the other hand, are named after the *only* symbol that can be emitted from them, for example C is the *only* symbol that is emitted from the first symbol state.

Symbol Emission Probabilities (B): The symbol emission probabilities are state specific. We assume that each non-gap state ($q_i \in Q_f \cup Q_s$) outputs only a single symbol, whereas gap states ($q_i \in Q_i \cup Q_u$) may output different symbols. The emission probability matrix is then given as: $B = \{b(q_i, v_m) = P(v_m|q_i), 1 \leq i \leq N \text{ and } 1 \leq m \leq M\}$, where $b(q_i, v_m) = P(v_m|q_i)$ is the probability of emitting symbol v_m in state q_i. $b(q_i, v_m)$ differs depending on whether q_i is a gap state or not. Since there is a chance that some symbols that do not occur in the training data may in fact be present in the test data, we assign them a very small probability of emission in the gap states.

Non-gap States: If $q_i \in Q_f \cup Q_s$, then $b(q_i, v_m) = 1$ for the distinct symbol that can be emitted for that state, and $b(q_i, v_m) = 0$, otherwise. For example, the first and second states are labeled by their emission symbol in Figure 1.

Universal Gap: For $q_i \in Q_u$ we have $b(q_i, v_m) = \left(\frac{freq(v_m)}{\sum_{v_m \in \Sigma} freq(v_m)}\right) \times 0.99 + c'$, where $c' = 0.01/M$. This means that v_m is emitted with probability proportional to its frequency in the training data. The c' term handles the case when v_m does not appear in the training set.

Gap States: If $q_i \in Q_i$, its emission probability depends on the symbol distribution mined by VGS. Let Σ_{q_i} be the set of symbols that were observed by VGS in the gap q_i. We have $b(q_i, v_m) = \left(\frac{\sum_{g \geq 1} freq_g(v_m, q_i)}{\sum_{v_m \in \Sigma_{q_i}} \sum_{g \geq 1} freq_g(v_m, q_i)}\right) \times 0.99 + c$, where $c = 0.01/|\Sigma_{q_i}|$.

Note that the above summations are for gap ranges $g \in [1, maxgap]$, since gap $g = 0$ is treated as a direct transition from one state to another. Note that the values 0.99 and 0.01 above arise from the pseudo-count approach used for previously unseen symbols. In our running example, for the symbol $v_m = C$ and the gap state g_4 between the states that emit A and B, we have the frequency of C as 2 out of the total number (5) of symbols seen in the gaps (see Section 3.1). Thus C's emission probability is $\frac{2}{5} \times 0.99 + \frac{0.01}{4} = 0.399$.

Gap Duration Probabilities (ρ): The probability of generating a given number of gaps from the gap states Q_i is given by the gap duration probability matrix: $\rho = \{\rho(q_i, g)|q_i \in Q_i, g \in [1, maxgap]\}$. Let q_i be the gap state between a state $q_x \in Q_f$ and a state $q_y \in Q_s$ corresponding to the sequence $s : v_x \rightarrow v_y \in F_2$. The gap duration probability is proportional to the frequency of observing a given gap value for s, i.e., $\rho(q_i, g) = \frac{freq_i^g(y)}{\sum_{g \in [1, maxgap]} freq_i^g(y)}$; and $\rho(q_i, g) = 1$ for $q_i \in Q \backslash Q_i$. In our running example, for the gap state g_4 between

the states that emit A and B, we have $\rho(g_4, 2) = \frac{2}{3} = 0.67$, since we twice observe a gap of 2, out of three occurrences.

Initial State Probabilities (π): The probability of being in state q_i initially is given by $\pi = \{\pi(i) = P(q_i|t = 0), 1 \leq i \leq N\}$, where

$$\pi(i) = \begin{cases} 0.99 \times \frac{\sum_{q_y \in Q_f} freq_i(y)}{T} & \text{for } q_i \in Q_f \\ 0.01 & \text{for } q_i \in Q_u \\ 0 & \text{for } q_i \in Q_i \cup Q_s \end{cases}$$

We use a small value for the Universal Gap state as opposed to the states in Q_f to accentuate the patterns retained by VGS while still providing a possibility for gaps after and before them.

3.3 Interpretation: Variable-Gap Viterbi

Once VOGUE is built, given a new test sequence of observations $O = o_1 o_2 \cdots o_T$, there is a need to interpret the sequence given the model. This problem is equivalent to finding the best sequence of states, i.e., the *most probable path*, through the VOGUE model λ, for the test sequence O. That is finding a sequence of states $\mathbf{q}_* = \{q_*^1, q_*^2, \cdots, q_*^T\}$ from the model λ such that: $\mathbf{q}* = \arg\max_{\mathbf{q}} P(\mathbf{q}|\lambda, O)$, over all such sequence of states $\mathbf{q}$. The algorithm that is most often used to solve this problem for biosequences, is the Viterbi algorithm [4]. Due to the unique structure of VOGUE, where gap states have a notion of duration, we adjusted Viterbi to take this into account. We call our new method Variable-Gap Viterbi (VG-Viterbi). For the lack of space, we omit the algorithmic details of VG-Viterbi.

4 Experimental Results and Analysis

In recent years, a large amount of work in biological sequence analysis has focused on methods for finding homologous proteins. Given a database of protein sequences, the goal is to build a statistical model so that we can determine whether a query protein belongs to a given family or not. HMMER [4], a profile HMM, is one of the state-of-the-art approaches to this problem that depends heavily on a good multiple sequence alignment. It models gaps, provided they exist in the alignment of *all* the training sequences. However, if a family of sequences has several overlapping motifs which may occur in different sequences, these sequences will not be aligned correctly and HMMER will not perform well. Here, we analyze the performance of VOGUE compared to HMMER and higher-order HMMs with various orders $k \in [1, 10]$.

Dataset: The data used in our experiments is a set of 9 families downloaded from the PROSITE (http://www.expasy.org/prosite) database of protein family and domains, namely, $PDOC00662$, $PDOC00670$, $PDOC00561$, $PDOC00064$, $PDOC00154$, $PDOC00224$, $PDOC00271$, $PDOC00397$, $PDOC00443$. We will refer

to these families as F_1, F_2, $\cdots$, F_9, respectively. The number of sequences in each family is, respectively: $N^1 = 45$, $N^2 = 225$, $N^3 = 85$, $N^4 = 56$, $N^5 = 119$, $N^6 = 99$, $N^7 = 150$, $N^8 = 21$, $N^9 = 29$. The families consist of sequences of lengths ranging from 597 to 1043 characters, taken from the alphabet of the 20 amino acids: $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$. Each family is characterized by a well-defined motif. Family F_1, for example, shares the consensus motif $[G] - [IVT] - [LVAC] - [LVAC] - [IVT] - [D] - [DE] - [FL] - [DNST]$, which has 9 components. Each component can contain any of the symbols within the square brackets. For example, for the second component, namely $[IVT]$, either I, V or T may be present in the sequences. We treat each PROSITE family as a separate class. We divided the data set of each family F_i, into two subsets: the training data N^i_{train} consists of 90% of the data, while the test data N^i_{test} contains the remaining 10%. For example, $N^1_{train} = 40$ and $N^1_{test} = 5$. There are a total of 103 test sequences across all families.

Evaluation and Scoring: We built three models for each family, namely VOGUE, HMMER and k-th order HMMs, using the training set of that family. We score the test sequences against the model for each of the nine families, and after sorting the scores in decreasing order, we use a threshold on the scores to assign a sequence to a given family.

For evaluation of the classifiers, we use Receiver Operating Characteristic (ROC) curves [5], that represent the relationship between the false positive rate and true positive rate across the full spectrum of threshold values. Further, we plot the Area Under the Curve (AUC), to evaluate the goodness of the classifiers. The AUC is calculated using the following equation [5]: AUC $= \frac{1}{pn} \sum_{i=1}^{p} \sum_{j=1}^{n} \varphi(R_i, R_j)$. Here $N_{test} = n + p$ is the number of test sequences, p is the number of sequences from a given class and n is the number of sequences that don't belong to the class. These sequences are ranked based on their score from 1 to N_{test}, assigning 1 to the test sequence with the highest score and N_{test} to the one with the lowest score. R_i, $i = 1 \cdots p$ represent the rankings of the p sequences and R_j, $j = 1 \cdots n$ represent the rankings of the n sequences and $\varphi(R_i, R_j) = 1$if $R_i < R_j$, or else $\varphi(R_i, R_j) = 0$. AUC for each class is calculated separately, by treating each class as p, and the remaining as n.

We score the test sequences by computing the log-odds score, i.e., the ratio of the probability of the sequence using a given model, to the probability of the sequence using a **Null** model, given as follows: Log-Odds(seq) $= \log_2 \left(\frac{P(seq/Model)}{P(seq/\mathbf{Null})} \right)$. $P(seq/Model)$ is computed using the Viterbi algorithm that computes the most probable path through the model, as Viterbi is the default method used for scoring in HMMER. The **Null** model is a simple one state HMM that emits the observations (the amino acids) with equal probability $(1/|\Sigma|)$. Since we have 20 amino acids, the emission probability for each symbol is 1/20. The log-odds ratio measures whether the sequence is a better match to the given model (if the score is positive) or to the null hypothesis (if the score is negative). Thus, the higher the score the better the model.

4.1 Comparing VOGUE, HMMER and k-th Order HMMs

We built VOGUE state machines with different values of *minsup* corresponding to 50%, 75% and 100% of the number of instances in the training data, and *maxgap* (10, 15, 20, 25, 30) but with the constant $k = 2$ for the length of the mined sequences in VGS. We then choose the best set of parameters and fix them for the remaining experiments. For HMMER, we first need to align the training sequences using CLUSTAL-W (http://www.ebi.ac.uk/clustalw). We then build a profile HMM using the multiple sequence alignment and compute the scores for each test sequence using HMMER, which directly reports the log-odds scores with respect to the **Null** model mentioned above. We also built several k-th order HMMs for various values of k using an open-source HMM software (http://www.cfar.umd.edu/∼kanungo/software). We tried different values for the number of states ranging from the size of the protein alphabet (20) to roughly the size of VOGUE (500) and HMMER (900). A k-th order HMM is built by replacing each consecutive subsequence of size k with a unique symbol. These different unique symbols across the training and test sets were used as observation symbols. Then we model the resulting sequence with a regular 1st order HMM.

Table 3. Test Sequence Log-Odds Scores for VOGUE, HMMER and k-th Order HMMs

Seq	VOGUE	HMMER	$k=1$ $M=20$	$k=2$ $M=394$	$k=4$ $M=17835$	$k=8$ $M=20216$	$k=10$ $M=19249$
S_1	**7081**	912.4	-4×10^3	-1.3×10^4	-2.3×10^4	-2×10^4	-2.6×10^4
S_2	**7877**	155	-3.4×10^3	-1.3×10^4	-2.2×10^4	-1.9×10^4	-2.9×10^4
S_3	**2880**	−345	-2.2×10^3	-1×10^4	-1.8×10^4	-1.6×10^4	-2.3×10^4
S_4	**5763**	9.8	-4.7×10^3	-1.5×10^4	-2.4×10^4	-2.2×10^4	-3.0×10^4
S_5	**5949**	−21.3	-4.7×10^3	-1.5×10^4	-2.4×10^4	-2.2×10^4	-3.1×10^4

Score Comparison: We first compare VOGUE with k-order HMMs and HMMER. Table 3 shows the comparison on the 5 test sequences for family F_1 when scored against the model for F_1. For VOGUE we used $minsup = 27(75\%)$ and $maxgap = 20$. For k-order HMMs we tried several values of the order k (shown as $k = 1$, $k = 2$, $k = 4$, $k = 8$ and $k = 10$) in the table with 20 states for each k-th order HMM. The number of observations M for the $k = 1$ case was set to 20 since it is the number of amino acids. $M = 394; 17835; 20216; 19249$ were the number of observations used for, respectively, $k = 2; 4; 8; 10$. These values were obtained from a count of the different new symbols used, as described earlier, for each value of k. The best score for each sequence is highlighted in bold. In Table 3, we find that k-th order HMMs were not able to model the training sequences well. All their scores are large negative values. HMMER did fairly well, which is not surprising, since it is specialized to handle protein sequences. However, for all the 5 test sequences VOGUE vastly outperforms HMMER. This is a remarkable result when we consider that VOGUE is completely automatic

and does not have explicit domain knowledge embedded in the model, except what is recovered from relationship between symbols in the patterns via mining.

Time Comparison: In Table 4, we show the execution times for family F_1. The time for VOGUE includes the mining by VGS, and for HMMER, the alignment by CLUSTAL-W. We can see that VOGUE's execution time is in general much better than HMMER and is also better than higher-order HMMs (except for $k = 1$). Thus, not only is VOGUE more accurate in modeling the input, but it also executes faster.

Table 4. Run Times

VOGUE	HMMER	$k = 1$	$k = 2$	$k = 4$	$k = 10$
$4.6s$	$34.42s$	$2s$	$5.29s$	$6.40s$	$11.46s$

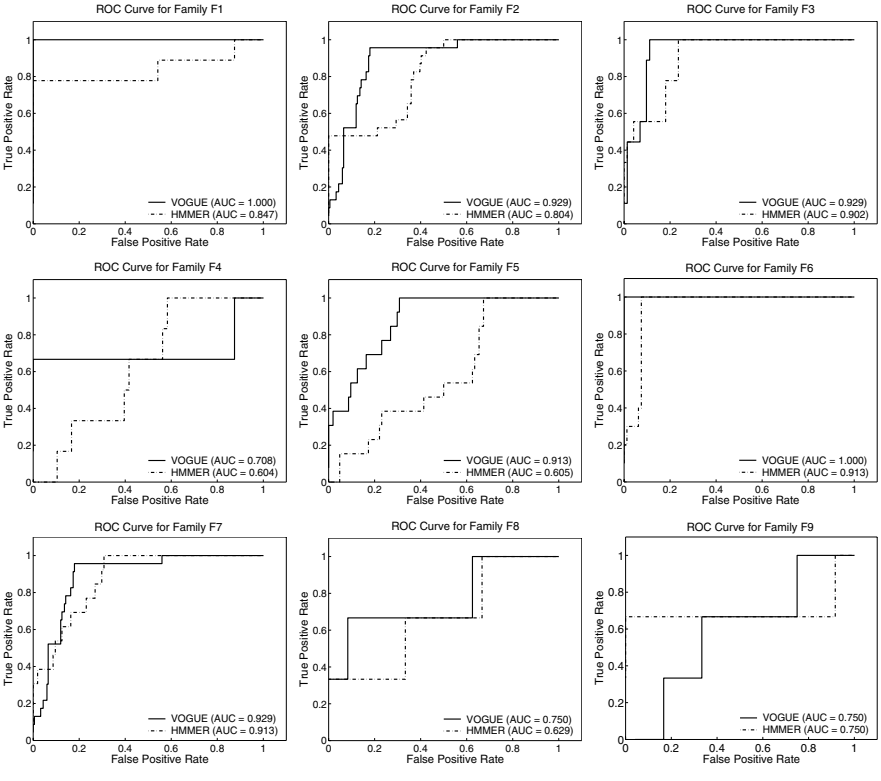


Fig. 2. ROC Curve of VOGUE and HMMER for the 9 families

Full Comparison (ROC Curves and AUC): Figure 2 presents the ROC curves of the 9 families generated from all the test sequences. Here we focus on comparing HMMER and VOGUE, since k-th order HMMs gave highly negative scores for all the test sequences. The ROC curves represent the trade-off between coverage (TPR on the y-axis) and error rate (FPR on the x-axis) of a classifier.

A good classifier will be located at the top left corner of the ROC graph. A trivial rejector will be at the bottom left corner of the ROC graph and a trivial acceptor will be at the top right corner of the graph. Each one of the graphs in Figure 2 has two ROC curves for VOGUE and HMMER, respectively, for different threshold values. The total AUC for the two methods is given in the legend. VOGUE was run with parameter typical values of $minsup = 75\%$ and $maxgap = 20$; there were some minor variations to account for characteristics of different families. The ROC curves of all the families show clearly that VOGUE improved the classification of the data over HMMER because the AUC of VOGUE is constantly higher than HMMER. In the case of family F_9 the AUC of both VOGUE and HMMER were comparable. In two cases, for families F_1 and F_6, the AUC was 1 for VOGUE showing that VOGUE was able to capture the patterns of those families perfectly. Moreover, in 6 out 9 families the AUC for VOGUE was higher than 0.9 as opposed to HMMER whose AUC was greater than 0.9 in only 3 out of 9 families. This again shows that VOGUE outperforms HMMER.

5 Conclusions and Future Work

One of the main contribution of VOGUE is that it can *simultaneously model* multiple higher-order HMMs. We showed experimentally on protein sequence data that VOGUE's modeling power is superior to higher-order HMMs, as well as a domain-specific algorithm HMMER. To generalize VOGUE for sequences of $k > 2$ (after VGS), a special topology will be needed to handle interleaving patterns. Furthermore, some patterns mined by VGS are artifacts of other patterns, for example, if $A \rightarrow B$ is frequent, then there is a good chance that $B \rightarrow A$ will be frequent as well. We need special pruning mechanisms to separate primary patterns from artifacts. Moreover, there are applications where there is not always an exact match for the subsequences to be mined. In future work we plan to allow for approximate matches for the mined sequences and states.

Acknowledgments. This work was supported in part by NSF CAREER Award IIS-0092978, and NSF grant EIA-0103708.

References

1. C. Antunes and A. L. Oliveira. Generalization of pattern-growth methods for sequential pattern mining with gap constraints. *Machine Learning and Data Mining in Pattern Recognition, Springer LNCS Vol. 2734*, pages 239–251, 2003.
2. M. Botta, U. Galassi and A. Giordana. Learning Complex and Sparse Events in Long Sequences. *European Conference on Artificial Intelligence*, 2004.
3. M. Deshpande and G. Karypis. Selective markov models for predicting web-page accesses. In *SIAM International Conference on Data Mining*, April 2001.
4. S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14:755–763, 1998.
5. P.F. Evangelista, M.J. Embrechts, P. Bonissone, and B. K. Szymanski. Fuzzy ROC curves for unsupervised nonparametric ensemble techniques. *IJCNN*, 2005.

6. S. Laxman, et al. Discovering frequent episodes and learning hidden markov models: A formal connection. *IEEE TKDE*, 17(11):1505–1517, Nov 2005.
7. A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. A data mining algorithm for generalized web prefetching. *IEEE TKDE*, 15(5):1155–1169, 2003.
8. J. Pitkow and P. Pirolli. Mining longest repeating subsequence to predict WWW surfing. In *2nd USENIX Symp. on Internet Technologies and Systems*, 1999.
9. L. Saul and M. Jordan. Mixed memory markov models: Decomposing complex stochastic processes as mix of simpler ones. *Machine Learning*, 37(1):75–87, 1999.
10. L.C. Schwardt and J.A. du Preez. Efficient mixed-order hidden markov model inference. In *Int'l Conf. on Spoken Language Processing*, oct 2000.
11. M. J. Zaki. Sequences mining in categorical domains: Incorporating constraints. In *9th Int'l Conf. on Information and Knowledge Management*, November 2000.
12. M. J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42(1/2):31–60, Jan/Feb 2001.

Don't Be Afraid of Simpler Patterns

Björn Bringmann, Albrecht Zimmermann, Luc De Raedt, and Siegfried Nijssen

Institute of Computer Science, Machine Learning Lab, Albert-Ludwigs-University Freiburg, Georges-Köhler-Allee 79, 79110 Freiburg, Germany
{bbringma, azimmerm, deraedt, snijssen}@informatik.uni-freiburg.de

Abstract. This paper investigates the trade-off between the expressiveness of the pattern language and the performance of the pattern miner in structured data mining. This trade-off is investigated in the context of correlated pattern mining, which is concerned with finding the k-best patterns according to a convex criterion, for the pattern languages of itemsets, multi-itemsets, sequences, trees and graphs. The criteria used in our investigation are the typical ones in data mining: computational cost and predictive accuracy and the domain is that of mining molecular graph databases. More specifically, we provide empirical answers to the following questions: how does the expressive power of the language affect the computational cost? and what is the trade-off between expressiveness of the pattern language and the predictive accuracy of the learned model? While answering the first question, we also introduce a novel stepwise approach to correlated pattern mining in which the results of mining a simpler pattern language are employed as a starting point for mining in a more complex one. This stepwise approach typically leads to significant speed-ups (up to a factor 1000) for mining graphs.

1 Introduction

Whereas initially the data mining community focused on mining simple pattern languages, such as itemsets, there has been a recent shift towards mining more and more complex pattern languages, such as sequences, trees and graphs [1,2,3]. This is often motivated by challenging applications domains such as chemoinformatics and network analysis, which can naturally be modeled as graphs. Whereas experience gained while mining simpler representations, such as the exploitation of the *apriori*-property [4], has been transferred into structured data mining, little research has been devoted to the trade-off between expressiveness of the pattern language and the performance of the pattern miner. Within the field of computer science, insight into the trade-off between expressiveness and performance has been of critical importance, and therefore also seems essential to the field of data mining and its applications.

The setting that we shall employ is that of correlated pattern mining, where one is given a dataset divided into two classes, and the aim is to find the k-best patterns expressed within a pattern language $\mathcal{L}$ according to some statistical criteria, such as significance, or information gain [5,6]. Correlated pattern miners typically employ a branch-and-bound technique. Because graph miners are today

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 55–66, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the most prominent representatives of structured data mining systems, and their application has been to a large extent targeted towards molecular applications in chemo-informatics such as structure activity relationship prediction [7], our experimental investigation targets such applications. The pattern languages $\mathcal{L}$ considered are those of *itemsets*, *multi-itemsets*, *sequences*, *trees*, and *connected graphs*. Even though there are many possible performance criteria in data mining, computational cost and predictive performance are certainly the most prominent ones, which we will therefore also adopt in our study. W.r.t. computational cost, the question is how the computational cost is affected by growing expressiveness of the language. While answering this question, we also introduce the technique of stepwise correlated pattern mining, which first finds the k best patterns in the simpler language, and then employs the score of the k-th best pattern as a bound while mining the more expressive language in a branch-and-bound algorithm. In some of our experiments, the stepwise approach leads to speed-ups of a factor 1000 for large molecular datasets. To gain insight into the second trade-off, which is concerned with predictive performance, we employ the typical approach of using the k best patterns as features in a classifier. This then allows us to construct a predictive model in the form of a decision tree, rule-set or naive Bayes model, and hence, to evaluate the predictive performance of the resulting model and thus, pattern language.

The paper is structured as follows: In the next section we introduce the notations used throughout the rest of the paper. In section 3, we describe correlation measures and the branch-and-bound approach used to solve the mining task. The experiments and their results are described and discussed in detail in section 4. In the last section we conclude and point towards related and future work.

2 Pattern Languages

In this section, we introduce the various pattern languages employed in our study, and discuss the relationship among them.

Definition 1 (Graphs). *An* undirected, labeled graph $\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda, \Sigma)$ *consists of a finite set* $\mathbb{V}$ *of vertices, a set* $\mathbb{E} \subseteq \{\{u, v\} | u, v \in \mathbb{V}, u \neq v\}$ *of edges, an alphabet* Σ *and a labeling function* $\lambda : (\mathbb{V} \cup \mathbb{E}) \to \Sigma$.

A graph $\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda, \Sigma)$ is called *connected* iff $\forall u, v \in \mathbb{V}$ there exists a sequence of vertices $\exists v_1, ..., v_n \in \mathbb{V}$ with $v_1 = u$ and $v_n = v$ and $\{v_x, v_{x+1}\} \in \mathbb{E}$.

Definition 2 (Trees). *A (free)* tree *is a connected graph with* $|\mathbb{V}| = |\mathbb{E}| + 1$.

Connected graphs that are not trees thus have cycles and are called *cyclic graphs*.

Definition 3 (Sequences). *A* sequence *is a tree (and hence a graph) where no vertex has more than two edges (i.e. no branches),* $\forall v \in \mathbb{V} : \left|\{\{v, u\} | u \in \mathbb{V}\}\right| \leq 2$.

Definition 4 (Multi-itemset). *A* multi-itemset *is a graph* $\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda, \Sigma)$ *where the set of edges is empty,* $\mathbb{E} = \emptyset$.

Definition 5 (Itemset). *A multi-itemset is called* itemset *iff* $\forall v_a, v_b \in \mathbb{V} : (v_a = v_b) \vee (\lambda(v_a) \neq \lambda(v_b))$.

Perhaps the definitions of multi-itemset and itemset are a bit unusual, but they are convenient for showing the relationship among the different pattern types. It is useful to use the following notation for the pattern languages: $\mathcal{L}_G$, the set of all graphs; $\mathcal{L}_C$, the set of all connected graphs; $\mathcal{L}_T$, the set of all trees; $\mathcal{L}_S$, the set of all sequences; $\mathcal{L}_M$, the set of all multi-itemsets; and $\mathcal{L}_I$, the set of all itemsets. From the definitions introduced above, it directly follows that

Proposition 1. $\mathcal{L}_I \subset \mathcal{L}_M \subset \mathcal{L}_G$ *and* $\mathcal{L}_S \subset \mathcal{L}_T \subset \mathcal{L}_C \subset \mathcal{L}_G$

At the same time, the subgraph isomorphism relation can be used as the *covers* and *generality* relation to structure the search:

Definition 6 (Graph Isomorphism). *Two Graphs* $\mathcal{G}(\mathbb{V},\mathbb{E},\lambda,\Sigma)$,$\mathcal{G}'(\mathbb{V}',\mathbb{E}',\lambda',\Sigma)$ *are called* isomorphic *if there exists a bijective function* $\varphi : \mathbb{V} \rightarrow \mathbb{V}'$ *such that:* $\forall v \in \mathbb{V} : \lambda(v) = \lambda'(\varphi(v)) \wedge\ \mathbb{E}' = \{\{\varphi(v_1), \varphi(v_2)\} | \{v_1, v_2\} \in \mathbb{E}\} \wedge\ \forall \{v_1, v_2\} \in \mathbb{E} : \lambda(\{v_1, v_2\}) = \lambda'(\{\varphi(v_1), \varphi(v_2)\})$

Definition 7 (Subgraph). *Given two graphs* $\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda, \Sigma)$, $\mathcal{G}'(\mathbb{V}', \mathbb{E}', \lambda', \Sigma)$, $\mathcal{G}'$ *is called a* subgraph *of* $\mathcal{G}$ *iff* $\mathbb{V}' \subseteq \mathbb{V} \wedge \mathbb{E}' \subseteq \mathbb{E} \wedge \forall v \in \mathbb{V}' : \lambda(v') = \lambda(v) \wedge \forall e \in \mathbb{E}' \lambda(e') = \lambda(e)$

A graph $\mathcal{G}'$ is subgraph-isomorphic to another graph $\mathcal{G}$ if and only if it has a subgraph $\mathcal{S}$ that is isomorphic to $\mathcal{G}'$. The subgraph isomorphism relation can be applied to the more specific pattern languages and results in a natural notion of coverage or generality for these languages as well.

3 Correlated Pattern Mining

A correlation measure compares the expected frequency of the joint occurence of a pattern and a certain class value to the observed frequency. If the resulting value is larger than a given threshold the deviation is considered statistically significant and there is evidence for a causal relationship between the pattern and the class.

	c_1	c_2	
ϕ	p	n	$p+n$
$\neg\phi$	$P-p$	$N-n$	$\|D\|-(p+n)$
	P	N	$\|D\|$

Fig. 1. A contingency table

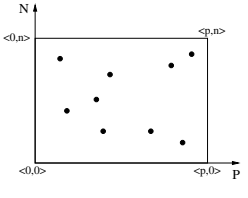


Fig. 2. Convex hull of future $\langle p\,, n\,\rangle$

We organize the observed frequencies in a contingency table, cf. Figure 1. Let ϕ be the pattern and c_1, c_2 the two classes. P and N denote the total number

of instances having class c_1, c_2, respectively, in dataset D. The frequencies of those two classes amongst the instances covered by the pattern are referred to as p and n. Since p and n are sufficient for calculating the value of a correlation measure on this table, we view these measures as real-valued functions on $\mathbb{N}^2$ for the remainder of this paper.

Correlation measures are neither monotone nor anti-monotone, hence the search becomes more difficult than in frequent pattern mining. However, if the correlation measure is convex, an upper bound on the score for the refinements of a given pattern can be calculated, which enables the use of an effective branch-and-bound algorithm.

Convex functions like χ^2 and *Information Gain* (see [5] for a proof) take their extreme values at the points forming the convex hull of their domain.

For a pattern inducing the tuple $\langle p, n\rangle$, specialization will decrease p, n, both, or none. This leads to points lying within the rectangle shown in Figure 2. Evaluating the correlation measure at $\langle p, 0\rangle$ and $\langle 0, n\rangle$ will give an upper bound on the value the measure can take for specializations of the current pattern. For an in-depth discussion of upper bound calculation we refer the reader to [5,6].

Correlated pattern miners now compute either the k best patterns within a language of patterns $\mathcal{L}$ or else finds all patterns within $\mathcal{L}$ whose correlation measure exceeds a certain threshold. Furthermore, to avoid redundancy in the resulting solution set, one typically employs patterns that are *free*, i.e., it is not allowed that two patterns ϕ_1 and ϕ_2 occur in the solution set for which ϕ_2 is a refinement of ϕ_1 and both patterns reach the same score.

The solutions to the first task can be conveniently modeled using

$$Th_k(\mathcal{L}, D) = \{\phi \in \mathcal{L} | free(\phi, D), \phi \textit{ among the k-best patterns w.r.t. } score(\phi, D) \textit{ in } \mathcal{L}\}$$

where D denotes the dataset under consideration, and *score* the correlation measure considered. Solutions to the second type of query are represented as

$$Th_>(\mathcal{L}, D, t) = \{\phi \in \mathcal{L} | free(\phi, D) \wedge score(\phi, D) > t\}$$

It will be convenient to combine the notations and introduce:

$$Th_k(\mathcal{L}, D, t) = \{\phi \in \mathcal{L} | free(\phi, D) \text{ and } score(\phi, D) > t \text{ and } \phi \textit{ is amongst the k-best patterns w.r.t. } score(\phi, D) \text{ in } \mathcal{L}\}$$

In the light of the hierarchies of languages introduced above, it is now instructive to look at some straightforward properties of these solution sets.

Proposition 2. *Whenever $\mathcal{L}_1 \subseteq \mathcal{L}_2$, then for all datasets D and thresholds t: $Th_>(\mathcal{L}_1, D, t) \subseteq Th_>(\mathcal{L}_2, D, t)$.*

This property actually states that whenever $\mathcal{L}_1 \subseteq \mathcal{L}_2$, it will be the case that solutions found within $\mathcal{L}_1$ will also be solutions within the more expressive pattern language $\mathcal{L}_2$. Another useful property is

Proposition 3. *Whenever $\mathcal{L}_1 \subseteq \mathcal{L}_2$, then for all datasets D: $Th_k(\mathcal{L}_2, D) \cap \mathcal{L}_1 \subseteq Th_k(\mathcal{L}_1, D)$.*

It states that the solutions of a top k query in a larger language $\mathcal{L}_2$ may contain some top k solutions from a less expressive language $\mathcal{L}_1$.

These properties actually motivate the stepwise correlated pattern mining algorithm, which we will now sketch. Algorithm 1 assumes a given hierarchy of pattern languages $\mathcal{L}_1 \subset \mathcal{L}_2 \subset \mathcal{L}_3 \ldots \mathcal{L}_n$, a dataset D, and a value for k. The goal is to find $Th_k(\mathcal{L}_n, D)$.

Algorithm 1. Stepwise Correlated Pattern Mining.

$t_0 := -\infty$;
for i = 1 to n **do**
  compute $Th_k(\mathcal{L}_i, D, t_{i-1})$
  $t_i := \min_{p \in Th_k(\mathcal{L}_i, D, t_{i-1})} score(p, D)$
return $Th_k(\mathcal{L}_i, D, t_i)$

The idea underlying the stepwise algorithm is that one first searches for the k best patterns in the simpler language $\mathcal{L}_i$, and then records the score of the k-th best solution found in $\mathcal{L}_i$. This score must be lower than or equal to the score of all solutions in $Th_k(\mathcal{L}_{i+1}, D)$, and can therefore be used as a threshold when searching for the solutions at the next level. Given that correlated pattern miners employ a branch-and-bound algorithm, this is likely to result in additional pruning when searching $\mathcal{L}_{i+1}$. This process is then iterated until the final language $\mathcal{L}_n$ is considered.

It is easy to see that the stepwise correlated pattern mining algorithm will produce exactly the same results as directly computing the $Th_k(\mathcal{L}_n, D)$. The question however is whether this stepwise technique is more efficient than the direct approach. In the experimental section, we shall provide evidence that this typically is the case for the hierarchies of pattern languages considered, and that the speed-up can be significant (up to a factor of about 1000).

4 Experimental Evaluation

In this section, we experimentally answer the following questions:

Q1 Does the stepwise correlated pattern miner speed up the correlated pattern mining process for the mining of graphs?
Q2 How does the expressiveness of the pattern language influence the running times of the correlated pattern miner?
Q3 How does the expressiveness of the pattern language influence the predictive performance of the models learned using correlated patterns as features ?

4.1 Experimental Set-Up

In all experiments, we employed a correlated pattern miner to compute$Th_k(\mathcal{L}, D)$ for the values $k = 1, 10, 100$ and 1000, and the languages $\mathcal{L} = \mathcal{L}_I, \mathcal{L}_M, \mathcal{L}_S, \mathcal{L}_T$ and $\mathcal{L}_C$. For the mining step we implemented two correlated pattern miners. For

the itemsets and multi-itemsets a simple APRIORI-SMP [5]-like implementation was used. For sequences, trees and graphs a modified GSPAN [2] implementation, able to mine correlated patterns, was employed. The correlation measure used was χ^2 and the starting minimum threshold for the first step in the stepwise mining procedure and all direct runs was 3.84.

To guarantee that we find all k-best free patterns, it is essential that for every large graph, all its subgraphs are enumerated first. Otherwise, it might happen that the addition of a small subgraph to a set of k-best patterns necessitates the removal of several k-best supergraphs, and, consequently, one could not guarantee that we find all k-best free patterns. It can be shown however that the enumeration schemes that we are using (both the stepwise approach, and GSPAN's method for enumerating graphs) have this desirable property. Furthermore, the properties of the DFS-Encoding used for graphs in GSPAN allowed in a straightforward manner to restrict the mining process to only trees (sequences) by not allowing structures with cycles (branches).

All running time experiments were performed on a 2.8GHz Machine with 2GB of main memory running Linux.

The two evaluation criteria employed were, on the one hand, the computation time needed to compute $Th_k(\mathcal{L}, D)$ and, on the other hand, the predictive accuracy. As directly evaluating the predictive accuracy of a correlated pattern is typically not very interesting, we rather evaluate the predictive accuracy of a set of patterns $Th_k(\mathcal{L}, D)$ indirectly. This is realized by employing these patterns as features in binary vectors describing the datasets to the WEKA toolkit. We then employed WEKA's [8] implementations of RIPPER [9], C4.5 [10], a SUPPORT VECTOR MACHINE, and NAÏVE BAYES to build classifiers whose accuracies can be measured. All accuracy estimates were obtained using ten-fold cross-validation, and the results were compared against each other w.r.t. significance. The WEKA experimenter environment was used for this task since accuracy averaging and significance testing is automated in this tool.

4.2 Datasets

For the experimental evaluation, we used four different real-world datasets, namely the *NCI HIV* dataset [3], a biodegradability dataset [11], and two mutagenicity datasets [12,7]. All these datasets contain a number of graphs describing chemical compounds. Atoms are represented as vertices and labeled with the atom type, whereas bonds between atoms are represented as edges, also labeled with the type of edge (which can be single bond, double bond or aromatic bond).

To answer the first two questions, we performed running time experiments on the **HIV** and on the **Mutagenicity I** datasets. Due to the small size of the other two datasets, running times measured are in the range of a few seconds at most, making differences unreliable. We used 5 different setups of the HIV dataset, namely *active vs. inactive* (CA vs CI), *active vs. moderate* (CA vs CM), *moderate vs. inactive* (CM vs CI), *active vs. moderate and inactive* (CA vs CMCI), and *active and moderate vs. inactive* (CACM vs CI). For each of the 6 setups, we mined the k-best patterns for $k = 1, 10, 100$, and 1000.

Table 1. Characteristics of the used datasets

Name	Total Size	Number of Classes	Class sizes
HIV	41768	3 (CA,CM,CI)	CA: 417, CM: 1069, CI: 40282
Biodegradability	328	2 (BD, NBD)	BD: 185, NBD: 143
Mutagenicity I	4337	2 (M,NM)	M: 2401, NM: 1936
Mutagenicity II	684	2 (M,NM)	M: 341, NM: 343

4.3 Stepwise Correlated Pattern Mining Q1

Our first question was whether the stepwise approach would speed up the mining process compared to the direct approach. The experiments performed show that the stepwise approach was up to 2800 times faster with an average of 202.81 ($\sigma = 600.66$) on all settings. Some of the results are shown in Table 2. The direct approach for $k = 1000$ in the *CS vs. CM* on the **HIV** dataset took more than 48 hours[1]! Figure 3 shows an explanation for the acceleration of the mining process. After evaluating only 600 sequences, the threshold reaches a value of 769. When mining trees (resp. graphs) directly, a similar threshold is reached after analyzing 2700 (resp. 4200) patterns. The quickly rising threshold enforced by mining stepwise allows a much more efficient pruning than the direct approach.

However, we observed some contrary cases that require further explanation. First, when comparing the direct and the stepwise approach on multi-itemset mining ($\mathcal{L}_M$), the stepwise approach does not help in terms of speed-up. Fortunately, it does not hurt either. Furthermore, on the **Mutagenicity I** dataset the stepwise approach was always slightly slower than the direct approach with the worst case for $k = 10$. Finally the direct mining of the 1000-*best* patterns on the *CA vs. CI* **HIV** dataset was 13.41% faster than the stepwise mining. In this case, the sequence-mining step could raise the threshold to 18.62 only which does not help much in the next mining step, especially considering the score of the 1000th-best pattern in $\mathcal{L}_C$ with 796.82. Furthermore, each step has to rebuild the *k-best* list since only the threshold is reused. This can be time consuming and might - as in this case - have the effect that the stepwise approach is slower than the direct approach. This could be improved by reusing not only the threshold obtained by a mining step, but the whole *k-best* list.

4.4 Running Time Q2

This second question considers the influence of the expressiveness of the pattern language on the runtime. More precisely: How are the running times distributed among the steps for mining $\mathcal{L}_S$ to $\mathcal{L}_T$ to $\mathcal{L}_C$? In principle, mining $\mathcal{L}_S$ could increase the threshold t_S to such a high level that mining $\mathcal{L}_T$ using t_S is faster than the mining of $\mathcal{L}_S$. The experiments in Table 3 show that this is never the case. The time spent in the tree mining step is on average 22 times as high as the sequence mining step. The graph mining step, on the other hand, is never

[1] Before it ran out of memory.

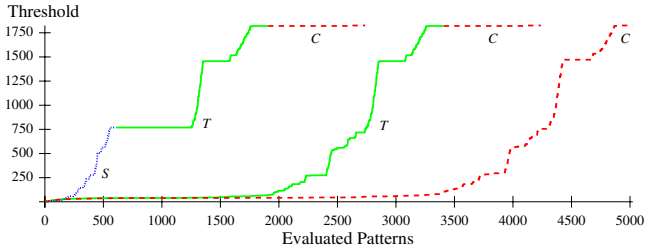


Fig. 3. Threshold vs. evaluated patterns for the stepwise and the direct approach on *HIV CA vs. CM* with $k = 100$. Mining $\mathcal{L}_S$ (dotted) followed by $\mathcal{L}_T$ (solid) and then $\mathcal{L}_C$ (dashed) evaluates much less patterns than mining $\mathcal{L}_T$ followed by $\mathcal{L}_C$. Mining $\mathcal{L}_C$ directly has to evaluate the most patterns.

much more expensive than the tree mining step. We also performed experiments on itemsets and multi-itemsets, which showed that itemsets were much faster than multi-itemsets. The results are not listed as other experiments showed that these features are not competitive.

Table 2. Running times of the stepwise and the direct approach for mining connected graphs ($\mathcal{L}_C$)

Approach	k	Muta I	CA vs CM	CACM vs CI	CA vs CI	CM vs CI	CA vs CMCI
direct	1000	0:05.16	>48h	7:48.21	0:19.54	4:07.15	0:23.51
stepwise	1000	0:06.39	0:03.57	5:11.19	0:22.59	3:02.15	0:23.15
direct	10	0:00.12	1:47.25	7:04.48	0:07.42	3:02.26	0:09.31
stepwise	10	0:00.22	0:00.14	0:05.63	0:01.03	0:07.18	0:01.10

Table 3. Running times of the stepwise approach for mining $\mathcal{L}_S, \mathcal{L}_T$, and $\mathcal{L}_C$, respectively. ($k = 1, 100$ are not listed).

Approach	k	Muta I	CA vs CM	CACM vs CI	CA vs CI	CM vs CI	CA vs CMCI
$\mathcal{L}_S$	1000	00:00.11	00:00.07	00:01.42	00:01.15	00:01.37	00:01.18
$\mathcal{L}_T$	1000	00:04.37	00:03.47	05:08.16	00:21.57	02:57.67	00:22.15
$\mathcal{L}_C$	1000	00:06.39	00:03.57	05:11:19	00:22.59	03:02.15	00:23.15
$\mathcal{L}_S$	10	00:00.03	00:00.04	00:00.56	00:00.36	00:00.57	00:00.38
$\mathcal{L}_T$	10	00:00.12	00:00.09	00:03.06	00:01.11	00:04.15	00:01.16
$\mathcal{L}_C$	10	00:00.22	00:00.14	00:06.13	00:01.03	00:07.18	00:01.10

4.5 Predictive Performance Q3

A typical usage of patterns found via a data mining approach is to employ them as features for describing instances to a propositional machine learning algorithm. Since mining more complex structures requires more time, as can be seen above, naturally the question arises whether this increase in mining effort

Table 4. Accuracy results on the HIV and Mutagenicity I datasets

(a) HIV *CA* vs *CM*

$\mathcal{L}_I$	$\mathcal{L}_M$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{L}_C$	k	Algorithm
72.88	74.16	76.78	**77.06**	**77.06**		J48
72.88	74.16	76.78	**77.06**	**77.06**	1	JRip
72.88	74.16	76.78	**77.06**	**77.06**		SVM
72.88	74.16	76.78	**77.06**	**77.06**		NB
72.88	74.16	**76.45**	76.28	76.28		J48
72.71	74.13	76.35	**77.06**	**77.06**	10	JRip
72.82	74.16	76.45	**76.85**	**76.85**		SVM
72.08	*73.02*	76.52	**76.65**	**76.65**		NB
73.05	74.16	**77.46**	77.06	77.06		J48
71.91	73.49	76.38	**77.06**	**77.06**	100	JRip
72.72	74.56	**77.09**	76.72	76.72		SVM
69.38	*72.21*	76.31	**76.58**	**76.58**		NB
73.05	*75.54*	**83.21**	*75.95*	*75.95*		J48
71.91	*74.84*	**81.29**	*76.52*	*76.42*	1000	JRip
72.72	*74.50*	**81.97**	*75.84*	*75.81*		SVM
69.38	*72.11*	**77.83**	76.65	76.65		NB

(b) Mutagenicity I

k	$\mathcal{L}_I$	$\mathcal{L}_M$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{L}_C$
	60.89	**61.15**	*58.20*	*58.40*	*58.40*
1	60.89	**61.15**	*58.20*	*58.40*	*58.40*
	60.89	**61.15**	*58.20*	*58.40*	*58.40*
	60.89	**61.15**	*58.20*	*58.40*	*58.40*
	61.40	*61.07*	**70.96**	*68.55*	*68.55*
10	*61.15*	*60.92*	**70.96**	*68.50*	*68.53*
	61.43	*60.45*	**70.19**	*68.52*	*68.49*
	61.17	*61.06*	**67.44**	*59.12*	*59.12*
	62.00	*65.67*	**76.37**	*70.94*	*70.79*
100	*61.33*	*65.11*	**74.06**	*70.74*	*70.89*
	62.17	*59.59*	**72.39**	70.23	70.17
	61.21	*60.24*	**70.08**	*65.67*	*65.90*
	62.00	*61.30*	**79.73**	*74.83*	*74.68*
1000	*61.33*	*60.89*	**76.40**	*72.23*	*71.66*
	62.17	*67.24*	**80.14**	*71.24*	*70.61*
	61.21	*60.80*	**71.70**	*57.61*	*57.33*

is matched by an increase of the quality of the description derivable by using the found patterns (**Q3**).

The first evaluated setting involved the HIV *active* and *moderately active* datasets. Accuracy estimates are show in Table 4(a). In this table, and all following ones, the best value an algorithm achieves for a given k is shown in **bold** numbers. All values that are significantly worse at $\sigma = 5\%$ are *italicized*. Itemsets never manage to capture enough information to be useful as features in this case and multi-itemsets also perform worse than the structured representations even though the difference is not significant except in the case $k = 1000$. As for the structured patterns - sequences, trees, and connected graphs - there is no significant difference for $k = 1, 10, 100$. For $k = 1000$, sequences significantly outperform trees and graphs except when used with the Naïve Bayes classifier which is overwhelmed by the number of features.

The second dataset was first used in a machine learning setting in [12]. Accuracy results on this dataset are summarized in Table 4(b) Except when only one feature is used, sequences are the most successful pattern type w.r.t. accuracy of the learners using those as features. The improvement in comparison to the other feature classes is significant except for the SVM when $k = 100$.

On the *biodegradability* dataset, there is no significant difference in accuracy for $k = 1$ and $k = 10$ as shown in Table 5(a). All representations more complex than pure itemsets perform very similarly to each other for the other two settings, with the notable differences that the SVM makes far better use of the sequence-type feature than of tree and graph features for $k = 100$ and that trees perform better than graphs in J48 and SVM for $k = 1000$. Finally, on the second mutagenicity dataset, for which results are reported in Table 5(b),

Table 5. Accuracy results on the Biodegradability and Mutagenicity II datasets

(a) Biodegradability

$\mathcal{L}_I$	$\mathcal{L}_M$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{L}_C$	k	Algorithm
67.05	**67.38**	65.25	65.25	65.25		J48
67.05	**67.38**	65.25	65.25	65.25	1	JRip
67.05	**67.38**	65.25	65.25	65.25		SVM
67.05	**67.38**	65.25	65.25	65.25		NB
65.51	68.43	**72.42**	72.27	72.27		J48
64.45	68.44	**74.56**	74.42	74.42	10	JRip
67.05	67.34	71.96	**72.43**	**72.43**		SVM
67.05	68.13	**74.09**	73.95	73.95		NB
65.51	72.58	**76.22**	71.96	71.81		J48
64.45	76.41	**77.89**	72.73	72.73	100	JRip
67.05	75.32	**80.04**	*71.96*	*71.81*		SVM
67.05	**69.04**	68.93	66.80	66.80		NB
65.51	73.45	74.10	**76.07**	71.20		J48
64.45	77.45	76.36	**78.95**	*72.42*	1000	JRip
67.05	74.89	76.08	**78.66**	*73.02*		SVM
67.05	**68.79**	64.65	63.13	62.98		NB

(b) Mutagenicity II

k	$\mathcal{L}_I$	$\mathcal{L}_M$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{L}_C$
	64.47	**64.91**	64.32	64.32	64.32
1	64.47	**64.91**	64.32	64.32	64.32
	64.47	**64.91**	64.32	64.32	64.32
	64.47	**64.91**	64.32	64.32	64.32
	65.86	*64.91*	**70.46**	**70.46**	**70.46**
10	*66.08*	*62.27*	**70.39**	**70.39**	**70.39**
	65.93	*62.79*	**70.39**	**70.39**	**70.39**
	64.62	64.47	**67.82**	**67.82**	**67.82**
	65.86	*64.39*	**70.90**	70.39	70.53
100	*66.08*	*64.75*	**71.48**	71.12	71.12
	65.93	*65.50*	**72.72**	71.71	71.78
	64.25	*66.08*	**72.07**	71.05	71.05
	65.86	*65.12*	71.63	71.55	**72.06**
1000	*66.08*	*67.87*	74.41	72.95	**74.48**
	65.93	*69.14*	**75.95**	74.99	75.13
	64.25	*64.77*	71.26	72.50	**72.87**

itemsets and multi-itemsets are significantly outperformed by the more complex representations. These in turn show no clear best or worst language class among themselves. The results of all experiments are surprising in that the use of more expressive pattern languages than $\mathcal{L}_S$ does not seem to pay off in terms of predictive accuracy. In all settings, sequences were at least as informative as trees and graphs when representing molecules. Yet sequences are much easier to handle and to compute than trees and graphs. On the other hand, the information stored in itemsets and multi-itemsets is typically not precise enough to be useful for a propositional learner. To gain more insight into the underlying reasons for these findings, we set up a further experiment in which we selected the $k = 1000$ best patterns in $\mathcal{L}_C \cup \mathcal{L}_M$ and classified them as (cyclic) graphs, trees, sequences, multi-itemsets and itemsets.

The two charts in Figure 4 show the distribution of the patterns, as well as the number of such patterns, their average score, and the standard deviation. Most of the experiments resulted in a chart similar to the one shown in Figure 4 (right) where the vast majority are trees, and the best scoring pattern is also a tree. In two exceptions like in Figure 4 (left), the multisets scored surprisingly well whereas in most of the other cases no single multiset or itemset ever appeared among the 1000 best. In two cases, the highest scoring structure was a sequence, and even though the sequences were much less frequent, they scored comparative to the trees. Furthermore, graphs (with cycles) usually had very low scores. A feasible explanation for these results might be in the fact that a graph contains far more sub-trees than sub-sequences and cyclic sub-graphs, and hence, that correlated pattern miners have to process much more trees (with similar scores) than sequences or cyclic graphs.

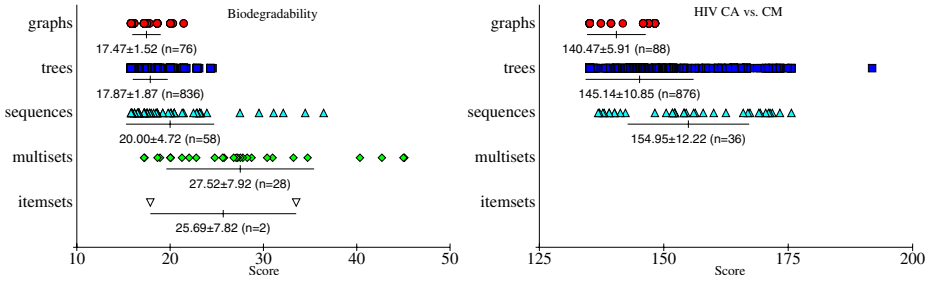


Fig. 4. Distribution of the $k = 1000$ best patterns for Biodegradability (left) and HIV CA vs. CM (right)

5 Conclusions

We have presented an empirical evaluation of the influence of the expressiveness of the pattern language on the performance of correlated graph miners. Perhaps the most surprising result of this study was that the use of more complex patterns such as cyclic graphs and trees does not necessarily lead to a better accuracy. Indeed, the best results obtained were by using sequential patterns, which are far easier to compute. A further result of our investigation was the introduction of a novel stepwise approach to correlated pattern mining, in which one first searches for k correlated patterns in a simpler language and then employs the score of the k-th best pattern as a lower bound for finding patterns at the next level of expressiveness. This stepwise approach led in virtually all cases to a significant speed-up, sometimes with a factor of up to 1000.

This work is related to the QUICKSTART approach by Nijssen and Kok [13] where a monotone constraint was considered. In this paper we study a branch-and-bound search using a convex constraint. This leads to further important differences. In our opinion, a constrained pattern mining algorithm consists of several elements: an algorithm that determines which candidates should be evaluated as it is not known yet if they satisfy the constraint, an algorithm that removes duplicate candidates and finally an algorithm that performs this evaluation. In the QUICKSTART approach, the last two steps were optimized. In this paper, we intend to optimize the first part. By considering a set of simple patterns first, we hope to find a threshold that allows for more pruning. Even though there were also speed-ups in the QUICKSTART approach, the magnitude was certainly not comparable to the factors obtained in correlated pattern mining. Finally, it would even be possible to combine both approaches. Concerning the influence of the predictive performance, related questions have arisen for instance in the kernel community, cf. [14], where different graph kernels take into account different types of information. Also, in the chemo-informatics community, cf. [15], it is often argued that one should take into account 3D information about the compounds in addition to the 2D graph structure. Doing this within our framework would be an interesting question for further research. It would

also be interesting to repeat our investigation in other domains than computational chemistry.

Acknowledgments. We are grateful to Andreas Karwath for providing the datasets. Furthermore, we like to thank the anonymous reviewers which helped us to improve the paper. The work was partially supported by the IQ project (EU grant IST-FET FP6-516169).

References

1. Zaki, M.: Efficiently mining frequent trees in a forest. In Hand, D., Keim, D., Ng, R., eds.: KDD. (2002) 71–80
2. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: ICDM. (2002) 721–724
3. Kramer, S., De Raedt, L., Helma, C.: Molecular feature mining in HIV data. In: KDD. (2001) 136–143
4. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB. (1994) 487–499
5. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: PODS. (2000) 226–236
6. Zimmermann, A., De Raedt, L.: Corclass: Correlated association rule mining for classification. In Suzuki, E., Arikawa, S., eds.: DS. (2004) 60–72
7. Helma, C., Cramer, T., Kramer, S., De Raedt, L.: Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. Journal of Chemical Information and Computer Systems **44** (2004) 1402–1411
8. Frank, E., Hall, M., Trigg, L.E., Holmes, G., Witten, I.H.: Data mining in bioinformatics using Weka. Bioinformatics **20** (2004) 2479–2481
9. Cohen, W.W.: Fast effective rule induction. In Prieditis, A., Russell, S.J., eds.: ICML. (1995) 115–123
10. Quinlan, J.R.: C4.5: Programs for Machine Learning. (1993)
11. Blockeel, H., Dzeroski, S., Kompare, B., Kramer, S., Pfahringer, B., Laer, W.V.: Experiments in predicting biodegradability. Appl. Art. Int. **18** (2004) 157–181
12. Kazius, J., Nijssen, S., Kok, J., Back, T., IJzerman, A.: Substructure mining using elaborate chemical representation. Journal of Chemical Information and Modeling **46** (2006) 597–605
13. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: KDD. (2004) 647–652
14. Horváth, T., Gärtner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: KDD. (2004) 158–167
15. Wale, N., Karypis, G.: Acyclic subgraph-based descriptor spaces for chemical compound retrieval and classification. Technical report, Univ. Minnesota (2006)

An Adaptive Prequential Learning Framework for Bayesian Network Classifiers

Gladys Castillo[1,2] and João Gama[1]

[1] LIACC, University of Porto, Portugal
[2] Department of Mathematics, University of Aveiro, Portugal
gladys@mat.ua.pt, jgama@liacc.up.pt

Abstract. We introduce an *adaptive prequential learning framework* for Bayesian Network Classifiers which attempts to handle the *cost-performance* trade-off and cope with *concept drift*. Our strategy for incorporating new data is based on *bias management* and *gradual adaptation*. Starting with the simple Naïve Bayes, we scale up the complexity by gradually increasing the maximum number of allowable attribute dependencies, and then by searching for new dependences in the extended search space. Since updating the structure is a costly task, we use new data to primarily adapt the parameters and only if this is really necessary, do we adapt the structure. The method for handling concept drift is based on the Shewhart P-Chart. We evaluated our adaptive algorithms on artificial domains and benchmark problems and show its advantages and future applicability in real-world on-line learning systems.

1 Introduction

We consider adaptive learning algorithms for Bayesian Network Classifiers (BNCs) in an *on-line* learning framework. In this framework data arrives at the learning system sequentially. The actual decision model must first make a prediction and then update the current model with new data. This philosophy about on-line learning frameworks has been exposed by Dawid in his *prequential* approach [6] for statistical validation of models. An efficient adaptive algorithm in a *prequential learning framework* must be able, above all, to improve its predictive accuracy over time while reducing the cost of adaptation. However, in many real-world situations it may be difficult to improve and adapt to existing changing environments. This problem is known as *concept drift*. In changing environments, learning algorithms should be provided with some control and adaptive mechanisms that try to adjust quickly to these changes.

The *Naïve Bayes* classifier (NB) is one of the most used classifiers in real-world on-line applications mainly due to its *effectiveness*, *simplicity* and *incremental nature*. NB simplifies learning by assuming that attributes are independent given the class. However, in practice, the independence assumption is violated which can lead to a poor predictive performance. We can improve the NB if we trade-off the *bias reduction* which leads to the addition of new attribute dependences,

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 67–78, 2006.
© Springer-Verlag Berlin Heidelberg 2006

and, consequently, to the estimation of more parameters, with the *variance reduction* by accurately estimating the parameters. Different classes of BNCs [5] attempt to reduce the bias of the NB by adding attribute dependences to the NB structure. Nevertheless, not always do the more complex BNCs outperform the NB. Increasing complexity decreases bias but increases the variance in the parameters. These issues are still more challenging in a *prequential framework*, where the training data increases with time. In this case, we should adjust the complexity of BNCs to suit the available data.

In this paper we present the Adaptive Prequential Framework for Supervised Learning, `AdPreqFr4SL`, which attempts to handle the *cost-performance* tradeoff and cope with *concept drift*. The `AdPreqFr4SL` strategy for incorporating new data is based on *bias management* and *gradual adaptation*. The motivations for bias control, along with some results of its application, were first presented in [3]. In the present work we have integrated more elaborated control tools for *bias management* with a method for *handling concept drift* based on *Statistical Quality Control* presented in [2] into the unified framework `AdPreqFr4SL`.

We chose the class of k-Dependence Bayesian Classifiers (k-DBC) [9] to illustrate our approach. A k-DBC is a Bayesian Network, which contains the structure of the NB and allows each attribute to have a maximum of k attribute nodes as parents. This class is very suitable for our proposal. By increasing k we can obtain classifiers that move smoothly along the spectrum of attribute dependencies. For instance, NB is a 0-DBC, TAN [5] is a 1-DBC, etc. Instead of using the learning algorithm proposed in [9] based on the computation of the conditional mutual information, we use a *hill-climbing* search procedure due to its obvious simplicity for computational implementation. The algorithm builds a k-DBC starting with an NB structure. Then it iteratively adds arcs between two attributes that result in the maximal improvements in a given score until there is no more improvement for that score or until it is no possible to add a new arc.

This paper is organized as follows. In the next section the `AdPreqFr4SL` is described. In Section 3 a compact overview of the conducted experiments that demonstrate the advantages of our adaptive approach is presented. In the last section we give some conclusions and lines for future work.

2 The Adaptive Prequential Learning Framework

The main environmental assumption that drives the design of the `AdPreqFr4SL` is that observations arrive at the learning system not at the same time, which allows the environment to change over time. Without loss of generality, we assume that at each time point data arrives in batches B. The main goal is to sequentially predict the classes of the next batch. Many adaptive systems employ regular update while new data arrives. The `AdPreqFr4SL`, instead, is provided with some controlling mechanisms that try to select the best adaptive actions according to the current learning goal. To this end, for each batch B of examples the current hypothesis is used to do *prediction*, the correct class is observed and

some performance *indicators* are assessed. Then, the indicator values are used to estimate the current system's *state*. Finally, the model is adapted according to the estimated state.

In the `AdPreqFr4SL` two performance indicators are monitored over time: the *batch error* Err_B (the proportion of misclassified examples in one batch) and the *model error* Err_S (the proportion of misclassified examples in the total of the examples that were classified using the same structure), in order to estimate one of the following states: [`S1`] - `IS IMPROVING`: the performance is improving; [`S2`] - `STOP IMPROVING`: the performance stops improving in a desirable tempo; [`S3`] - `CONCEPT DRIFT ALERT`: a first alert of concept drift is signaled; [`S4`] - `CONCEPT DRIFT`: there is a gradual concept change; [`S5`] - `CONCEPT SHIFT`: there is an abrupt concept change; [`S6`] - `STABLE PERFORMANCE`: the performance reaches a plateau. In the next subsections we present the adaptive actions and control strategies that we have adopted in the `AdPreqFr4SL` for handling the *cost-performance* trade-off and *concept drift*.

2.1 Cost-Performance Management

The adaptation strategy for handling *cost-performance* is based upon two main policies: *i*) *bias management* - starting with an NB structure, we scale up the model's complexity by gradually increasing k and then searching for new attribute dependences in the resulting search space; *ii*)*gradual adaptation* - we define four levels of adaptation so that increasing the level increases its cost. In the INITIAL LEVEL a new model is built using the simple NB. In the FIRST LEVEL only the parameters are updated with new data (optionally we can use the Iterative Bayes [7] for parameter refinement). In the SECOND LEVEL the structure is updated with new data. In the THIRD LEVEL, if it is still possible, k is increased by one, and the current structure is once again adapted.

The rationale is as follows. We initialize k-DBC to the simple NB by setting $k = 0$. Whenever new data arrives, we first try to improve the NB by adapting only its parameters. When there is evidence indicating that the performance of the NB stops improving in a desirable tempo, we start adapting the structure. Only in this case (for $k = 0$) do we move from the *first level* to the *third level* of adaptation: increment k by one and start searching a 1-DBC using the *hill-climber* search procedure only with arc additions. At this time point we must have more data available which allows the search procedure to find new 1-dependencies. Next, the algorithm continues to perform only parameter adaptation. Thus, whenever a new structure is found, the algorithm continues working from the *first level* of adaptation, that is, by performing only parameter adaptation, until there will be again evidence that the performance of the current hypothesis stops improving and this moves to the *second level*: update the current structure by searching for new attribute dependencies. At this stage and to correct from previous errors, the search procedure is also allowed to perform *arc deletions*. Only if the resulting structure remains the same, do we move to the *third level* of adaptation by incrementing k by one and continue searching for new dependencies, now in an augmented search space. For avoiding k to increase

unnecessarily, we recover the old value of k whenever the search procedure is not able to find new dependencies, thus keeping the original search space. Only if an abrupt concept drift is detected, do we come back to the *initial level* and build a new NB using the examples from a short-term memory (see next section). This adaptation process will continue until it is detected that it does not make more sense to continue adapting the model. However, we will continue monitoring the performance. If any significant change in the behaviour is observed, then we will once again to activate the adaptation procedures.

The *control policy* defines the criteria for tracking two situations: *i*) At which time point do we start adapting the structure?; *ii*) At which time point do we stop doing any adaptation? If it is detected that the performance of the current model no longer improves in a desirable tempo (the state **S2**), we start adapting the structure. If it is detected that the performance reaches a plateau (the state **S6**), we stop adapting the model. To detect the states **S2** and **S6**, we plot the values of successive model errors, $y(t) = Err_S^{(t)}$, in time order and connect them by a line, thus obtaining the *model-error learning curve* (`model-LC`). We consider that the state **S2** is met if: *i*) the `model-LC` starts *behaving well* [1], i.e, the curve is *convex* and *monotically non-increasing* for a given number of points; *ii*) its slope is *gentle*. Thus, whenever we start using a new structure we will wait until `model-LC` starts *behaving well* and shows only little improvements in the performance in order to trigger a new structure adaptation. If the structure does not change after adaptation, we once again look at the `model-LC` to detect whether it has already reached its *plateau* (i.e. **S6** is signaled).

The following question thus arises: *How does one verify whether the required criteria are met*? From all the explored methods, we empirically found that by using a method based on the geometrical properties of the `model-LC`, which analyzes the graphical behaviour of the most recent q points, we could more consistently determine *discrete convexity* and the *slope* of the `model-LC` taking into account the local variance. We obtained the best results by setting $q = 7$.

As illustrated in Figure 1 we construct a triangle T with the points p_1, p_4, p_7 and use its *signed area*, $\mathcal{A}(T)$, to test for *discrete convexity* [8]. The points p_1, p_4, p_7 are arranged in a *convex pattern* iff $\mathcal{A}(T)$ is positive. In this case the path $p_1 \rightarrow p_4 \rightarrow p_7$ is oriented *counterclockwise* around the triangle. Taking into account the local variance we consider a *convex pattern*, if $\mathcal{A}(T) > \delta_a$ where δ_a is a very small negative number (our tolerance for convexity). Then, we analyze

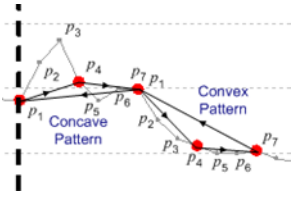


Fig. 1. The last seven points $p_1, p_2, ..., p_7$ in the current model-error curve are analyzed to determine the existence of a convex pattern and a decreasing trend

the angles formed between middle segments, $\angle_1 = \angle p_1, p_2, p_4, \angle_2 = \angle p_1, p_3, p_4$, $\angle_3 = \angle p_4, p_5, p_7$ and $\angle_4 = \angle p_4, p_6, p_7$ to determine if the remaining points are *almost colinear* given a tolerance δ_c, where δ_c is a very small positive number, that is, if $\sin(\angle_l) < \delta_c, \forall \angle_l, l = 1, 2, 3, 4$. We then use the Sen's slope estimator [10] for determining whether there is a *non-increasing trend* in these observed points. We consider a *non-increasing trend* if $\texttt{SenSlope}^{(q)} < \delta_s$ where δ_s is a very small positive number. We obtained satisfactory empirical results by setting $\delta_a = -0.0001$ (our tolerance for *convexity*), $\delta_s = 0.001$ (our threshold for *non-increasing* trend) and $\delta_c = 0.001$ (our tolerance for *colinearity*).

Thus, we consider that the points $p_1, p_2, \ldots, p_7$ are arranged in a *convex pattern* with a *non-increasing trend* and *gentle slope* if for a given positive small number ϵ_1, the threshold for the *gentle slope*, the following criterion is met:

$$\delta_a < \mathcal{A}(T) < \epsilon_1 \quad \wedge \ \sin(\angle_l) < \delta_c, \forall \angle_l, l = 1, 2, 3, 4 \ \wedge \texttt{SenSlope}^{(7)} < \delta_s \qquad (1)$$

We consider that the *stopping criterion* is met if given a positive small number ϵ_2, the threshold for the plateau, such that $\epsilon_2 < \epsilon_1$, the following criterion is met:

$$|\mathcal{A}(T)| < \epsilon_2 \ \wedge \ \sin(\angle_l) < \delta_c, \forall \angle_l, l = 1, 2, 3, 4 \qquad (2)$$

In addition, we use a heuristic based on the observation of the *batch error* before and after the adaptation, which has been demonstrated [3] to be efficient for an early detection of the point at which we should start adapting the structure. Whenever we obtain a decrease of the batch error after adaptation, we consider that the learner is still able to learn using the current structure. Otherwise, if for a pre-defined number of consecutive times, `maxTimes`, the batch error does not decrease after parameter adaptation we assume that increasing the number of training examples will not result in further improvements on the parameter estimates and signal the state [**S2**] .

Figure 2 illustrates the behaviour of the `model-LC` for one randomly generated sample of the *adult dataset* using batches of 100 examples. To serve as a baseline, we also plot the error rates obtained with the NB and with a 3-DBC (the class-model with best performance) induced from scratch at each learning step. During all the learning process the structure changed only five times. The graphical behavior of the model error neatly corresponds to the detected conditions which lead to a structure-adaptation action. The k value slowly increases from 0 to 3 until that the stopping criteria is met at $t = 120$ and the model is not further adapted with new data.

2.2 Using the P-Chart for Handling Concept-Drift

Concept drift refers to unforeseen changes in the distribution underlying the data that can also lead to changes in the target concept over time [11]. Several available concept drift trackers employ different approaches that include some control strategies in order to decide whether adaptation is really necessary because a concept change has occurred. To this end, a process that monitors the value of some performance indicators is implemented. If a concept drift is detected, some

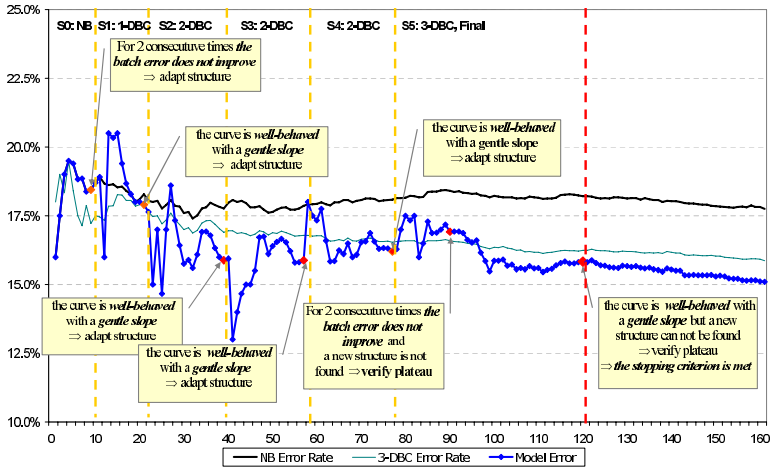


Fig. 2. Behavior of the `model-LC` for the adaptive algorithm. Vertical lines indicate the time points at which the structure changed. On top, the resulting structures with their corresponding k-DBC class-models are presented.

actions to adapt the model to these changes are taken, which usually lead to build a new model. Some concept drift trackers are also capable of recognizing the *extent* of concept drift. The term *concept drift* is more oftenly associated to gradual changes whereas the term *concept shift* defines abrupt changes.

We integrated into the `AdPreqFr4SL` a method for handling concept drift [2] based on a *Shewhart* `P-Chart` - an attribute control chart that monitors the proportion of a *dichotomous* count variable. We use the P-Chart for monitoring the batch error Err_B. The values $p(t) = Err_B^{(t)}$ are plotted on the chart in time order and connected by a line. The chart has a *center line* (`CL`), an *upper control limit* (`UCL`) and an *upper warning limit* (`UWL`). If the sample sizes are large (≥ 30) the *sample proportion* approaches the Normal distribution with parameters $\mu = p\,; \sigma = \sqrt{p(1-p)/n}$ (p is the population proportion). Therefore, the use of *three-sigma control* limits is a reasonable choice. Suppose that an estimate $\hat{p}$ is obtained from previous data. We can obtain the P-Chart's lines as follows: $\texttt{CL} = \hat{p}$; $\texttt{UCL} = \hat{p} + 3\sigma$; $\texttt{UWL} = \hat{p} + \alpha\sigma, 0 < \alpha < 3$. In this work, we set α to 2. To better follow the natural behaviour of the learning process we set the target value $\hat{p}$ to the minimum value of the current model error Err_S. We denote it by Err_{min}. Whenever a new structure S is found, Err_{min} is initialized to some big number. Then, at each time step if $Err_S^{(t)} + SErr_S^{(t)} < Err_{min}$ then Err_{min} is set to $Err_S^{(t)}$, where $SErr_S^{(t)}$ is its standard deviation.

Thus, at each time point t, $\hat{p}$ is set to Err_{min} and the P-Chart's lines are computed, accordingly. Then, it is observed where the new proportion $p(t) = Err_B^{(t)}$ falls on the `P-Chart`. If $p(t)$ falls above the `UCL`, a *concept shift* is signaled. If $p(t)$ falls between the `UCL` and the `UWL` for the first time, then a *concept drift alert* is signaled. Otherwise, if this situation occurs for two or more consecutive

times then a *concept drift* is detected. If $p(t)$ falls under `UWL` we assume that the learner is *in control* and then proceed to analyze the behaviour of the `model-LC` as described in the previous section.

The *adaptive strategy* for handling concept drift mainly consists of manipulating a *short-term memory* (`SHORT-MEMORY`) to store those examples that we suspect belongs to a new concept. If a *concept shift* is detected then all the examples from the `SHORT-MEMORY` are used to build a new NB classifier. Afterwards, the `SHORT-MEMORY` is cleaned for future uses. Whenever a *concept drift alert* or *concept drift* is signaled, the examples of the current batch are added to the `SHORT-MEMORY`. However, after signaling a *concept drift*, the new examples are not used to update the model in order to force a great degradation of the performance. This way the P-Chart will more quickly be able to recognize a concept shift and re-build the model. Algorithm 1 depicts the pseudo-code of the whole algorithm for learning k-DBCs in the `AdPreqFr4SL` that summarizes all the above described strategies for handling *cost-performance* and *concept drift*.

Algorithm 1. The algorithm for learning k-DBCs in `AdPreqFr4SL`

Require: A dataset $\mathcal{D}$ divided in batches of m examples, a `kMax` value for the maximum allowable k, the thresholds: `eps1` for the gentle slope and `eps2` for the plateau, the number of consecutive times `maxTimes` that Err_B does not decrease after parameter adaptation, a boolean variable `bIterativeBayes` for using Iterative Bayes or not, a scoring function `Score`$(S, \mathcal{D})$

Ensure: A classifier $h_{\mathcal{C}} = (S, \Theta_S)$ belonging to the class of k-DBCs

AdaptiveAction($h_{\mathcal{C}}$, `SHORT-MEMORY`, `INITIAL LEVEL`) {*build a new NB classifier*}
for each next batch B of m examples of $\mathcal{D}$ **do**
    predictions ← predict(B, $h_{\mathcal{C}}$)
    observed ← getFeedback(B) {get feedback}
    $p(t) \leftarrow Err_B^{(t)}, y(t) \leftarrow Err_S^{(t)}$ {asses current indicators}
    Add $(t, y(t))$ to `model-LC`
    state ← getState(p(t), P-Chart){concept drift detection using the P-Chart}
    if `state` is CONCEPT SHIFT **then**
        Add B to `SHORT-MEMORY`
        AdaptiveAction($h_{\mathcal{C}}$, `SHORT-MEMORY`, `INITIAL LEVEL`) {*build a NB classifier*}
        Clean `SHORT-MEMORY`
    else if `state` is CONCEPT DRIFT ALERT ∨ CONCEPT DRIFT **then**
        Add B to `SHORT-MEMORY`
    else
        Clean `SHORT-MEMORY`
        // *if state is* IN CONTROL *then observe the* `model-LC`
        if `model-LC` is Convex-NonIncreasing-with-GentleSlope(`eps1`) **then**
            state ← STOPS IMPROVING {conditions 1 are met}
        else
            state ← IS IMPROVING
    if `state` IS IMPROVING ∨ CONCEPT DRIFT ALERT **then**
        AdaptiveAction($h_{\mathcal{C}}$, B, `FIRST LEVEL`, `bIterativeBayes`){*update parameters*}
        if consecCounter($Err_B^{t_{\texttt{AFTER-ADAP}}} \geq Err_B^{t_{\texttt{BEF-ADAP}}}$) = `maxTimes` **then**
            `state` ← STOP IMPROVING
    if `state` STOPS IMPROVING **then**
        if $k > 0$ **then** AdaptiveAction(k-DBC, B, `SECOND LEVEL`,...) {*update structure*}
        if (not change(S) ∧ $k <$ `Maxk`) ∨ k= 0 **then**
            AdaptiveAction($h_{\mathcal{C}}$, B, `THIRD LEVEL`,k, ...) {*increment k; continue searching*}
        if not change(S) **then**
            // *verify the stopping criterion*
            if `model-LC` Has-Plateau(`eps2`) **then**
                stopAdapting ← TRUE; `state` ← STABLE PERFORMANCE
end for
return $h_{\mathcal{C}}$

3 Experimental Evaluation

We carried out a series of experiments for evaluating the `AdPreqFr4SL` for k-DBCs, using both, artificially generated datasets and benchmark problems from the UCI repository. Due to space limitations, we here provide only an overview of all the conducted experiments and results. A complete description is given in [4]. We evaluated two versions of adaptive algorithms, `Adap1` and `Adap2`, using Algorithm 1. `Adap2` additionally implements Iterative Bayes (IB). We compared `Adap1` and `Adap2` against NB and several k-DBCs (varying k) induced from scratch, i.e., in each learning step, a *batch hill-climber* learning procedure was used to learn a k-DBC from all seen examples. Since learning from scratch use all the data provided so far, this approach for updating the classifier is essentially optimal in terms of the quality of the hypotheses it can induce. We set `kMax=5` and `maxTimes`=2. The thresholds `eps1` (for the gentle slope) and `eps2` (for the plateau) were set according to the domain's complexity. Intuitively, we choose lower thresholds for more complex domains. All the results were obtained as average values over 10 generated samples. Here we present the results using the Bayesian score (the marginal likelihood). In [3,4] we give a more in depth study comparing the performance for different scores.

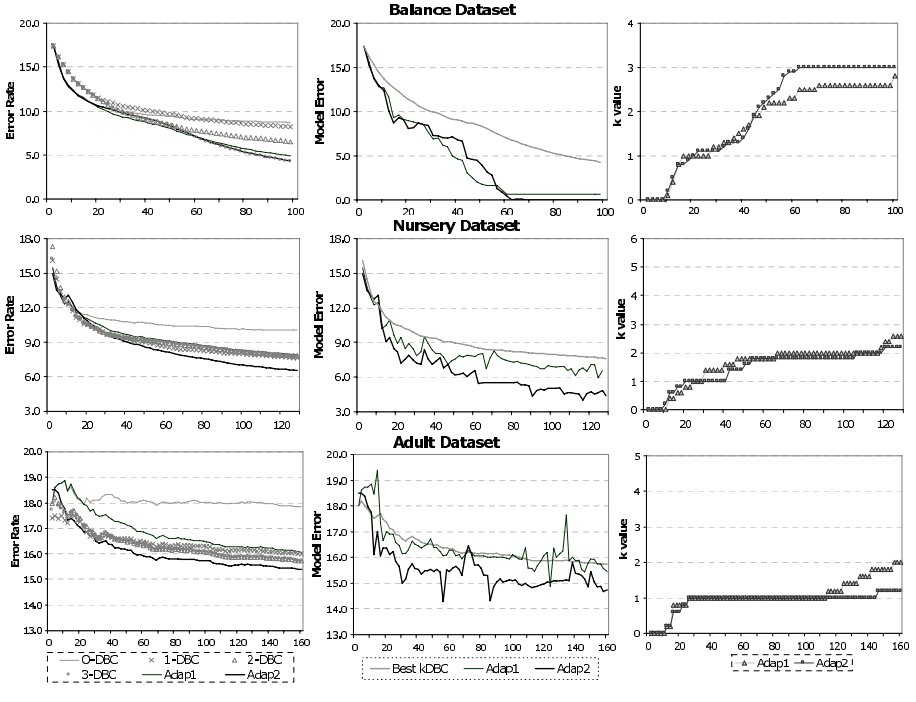


Fig. 3. Error Rate, Model Error and k-values for UCI's datasets

3.1 Evaluation with Benchmark Problems

We here only show the results for three selected datasets from the UCI repository: *balance*, *nursery* and *adult*. We set `eps1`=0.01, `eps2`=0.001 for *balance* and *nursery* and `eps1`=0.001, `eps2`=0.0001 for *adult*. Figure 3 compares the performance of Adap1 and Adap2 against NB and several k-DBCs induced from scratch at each time point. In most cases, Adap1 approaches the performance of the best k-DBC and Adap2 outperforms Adap1. For the *balance* dataset, the Err_S approaches 0 and k approaches 3, thus evidencing that Adap1 and Adap2 were able to find structures that represent the existing strong degree of attribute dependences.

Table 1 helps us to evaluate the *performance*, *complexity* and *cost of adaptation* per dataset at the last learning step. The results show that both, Adap1 and Adap2, are able to perform a more artful *cost-performance* trade-off than *non-adaptive* versions. The reduction of the cost of updating is evident if we compare the small number of adaptations performed on the structure by Adap1 and Adap2 in contrast to the great cost of searching for a new structure at each learning step. The number of times the structure really changed in the case when a search procedure was invoked at each point time is very small (e.g. for the *adult* this proportion is 8.6/160) thus evidencing that it is more appropriate to perform adaptations on the structure when there is some accumulated data and the search procedure is able to find new dependences. Although both, Adap1 and Adap2, show a desirable behaviour, results evidence that Adap2 ensures the best *cost-performance* trade-off in these three particular domains: the number of structure adaptations and the resulting error are smaller. We also observed that the increasing slope of the k value using Adap2 is more gradual, specially for more complex domains, thus inducing less complex classifiers than Adap1. Adap2 can get trapped in less complex structures while reducing the bias on the parameter estimates [7]. By using `AdPreqFr4SL` with IB, specially for more complex domains, we can better trade-off the reduction of the bias resulting

Table 1. Analysis of the Final Performance, Complexity and Cost of Adaptation per dataset. The column "`Last` Err_B" shows the error of the last batch of examples, which was not used to update the classifier. The column "`♯ Add. Arcs`" shows the final number of arcs added to NB. The column "`♯ Str.Adap.`" shows the total number of times that a structure-adaptation action was executed. The column "`♯ Str.Chang.`" shows the number of times the structure really changed over time.

	Balance				**Nursery**				**Adult**			
k-DBCs	Last Err_B	♯ Add. Arcs	♯ Str. Adap.	♯ Str. Chang.	Last Err_B	♯ Add. Arcs	♯ Str Adap.	♯ Str Chang.	Last Err_B	♯ Add. Arcs	♯ Str Adap.	♯ Str Chang.
NB	8.10	–	–	–	12.00	–	–	–	16.80	0.0	–	–
1-DBC	5.80	3.0	100	11.2	5.80	5.4	128	6.4	14.60	12.0	160	8.6
2-DBC	4.60	5.0	100	17.6	7.40	8.0	128	8.8	14.60	18.0	160	13.4
3-DBC	0.00	6.0	100	18.5	7.20	9.0	128	9.8	14.60	18.0	160	13.2
Best	0.00	6.0	100	18.5	5.80	5.4	128	6.4	14.60	12.0	160	8.6
Adap1	0.30	5.6	4.7	3.2	6.80	8.0	18.8	6.0	13.60	16.8	4.0	3.2
Adap2	0.00	6.0	4.2	3.8	4.40	7.0	11.6	5.2	13.20	12.2	2.6	2.4

from the assumptions of attribute independence with the reduction of the bias resulting from the *estimation error* by also improving the parameter estimates.

3.2 Evaluation with Generated Concept Shift and Drift Scenarios

Five *concept shift scenarios* (CSSs) and five *concept drift* scenarios (CDSs) were generated using randomly generated k-DBCs with 9 binary attributes and a binary class node for $k = 1, 2, 3, 4, 5$. Both, CSSs and CDSs represent a sequence of *five* different learning contexts, associated to different generative k-DBCs. Whereas k remains constant in a CSS, we used k-DBCs of increasing k for generating a CDS (a 1-DBC for the first context, a 2-DBC for the second one, etc.). In CSSs we simulated *four abrupt concept changes* by forcing the underlying k-DBC to change after every 2000 examples. We used batches of 100 examples for CSSs and batches of 50 examples for CDSs. In CDSs we simulated *four gradual changes* by setting the parameters of a simulation procedure [11]: $t_1 = 37, t_2 = 77, t_3 = 117, t_4 = 157$ (the time points at which the concept begins to drift), $\Delta = 300$ (the drift rate) and $\alpha = 3/4$ (each 3 examples of the old concept appears one example of the new concept). We set `eps1`=0.05 and `eps2`=0.005 for artificial datasets.

Figure 4 illustrates the adaptive and control strategies in one of generated CDS. In the *first drift phase* (between t=37 and t=43) the P-Chart detected two concept shifts and a new NB was built using the examples of the current batch. In the *second drift phase* (between t=77 and t=83) almost all the points fell above the UWL but very close to the UCL. The P-Chart signaled concept drift and the adaptation process was temporarily stopping to force the Err_B to jump outside the UCL. Later, at t=83, when a concept shift was detected, all the examples stored in the `SHORT-MEMORY` were used to build a new NB. For

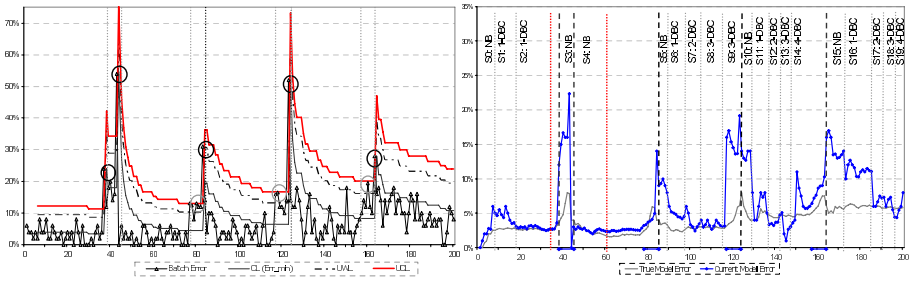


Fig. 4. The P-Chart(on the left) and the model error Err_S (on the right) for a generated CSS. Parallel light-grey dotted lines on the P-Chart indicate the beginning and the end of each drift phase. Vertical light-grey dotted lines and black dashed lines on the model-error's figure indicate the time points at which the current structure was adapted or rebuilt, respectively; and vertical dark-grey dotted lines, the time points at which the adaptation process was stopped.In the top, the resulting structures with their corresponding k-DBC class-models are presented.

the remaining drift phases our detection method using P-Chart also worked as expected. As a result, the structure was rebuilt five times, at time points that belong to the drift phases. Note that the complexity of the induced k-DBCs increased from context to context: in the first context the resulting k-DBC is a 1-DBC, in the third - a 3-DBC, in the fourth - a 4-DBC, in the last context it is a 4-DBC too (searching for more complex structures can require more training data). Only in the second context the NB structure was not modified since the adaptation process was stopped early. However, the model error showed a good behaviour in this context.

Figure 5 compares the performance over time of all the algorithms in the CSS associated to a 2-DBC (CS-II) and one CDS scenario (CD-I). Results evidence that significant improvements in the performance are achieved by using adaptive algorithms instead of their non-adaptive versions. After a concept drift occurs, the performance of all the algorithms suffers a significant deterioration. However, Adap1 and Adap2 show a good *recoverability* capability and are able to control the performance, trying to improve it back to a level, that even approaches the performance of the *true model*. In drift phases, the k value falls to 0, which evidences that a concept shift has been detected and a new NB has been built. Results also evidence that adaptive algorithms approach the appropriate class-model associated to each learning context. The k value approaches 2 for all the learning contexts in CS-II while it increases from context to context in CD-I.

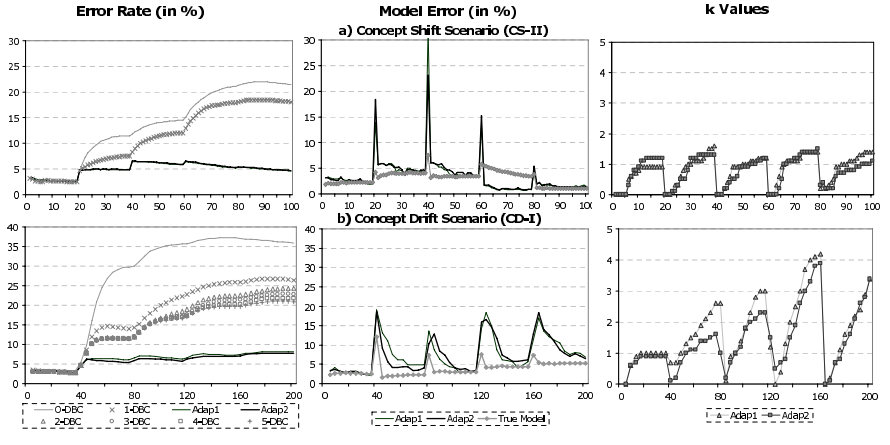


Fig. 5. Error Rate, Model Error and k-values for CS-II and CD-I

4 Conclusions and Future Work

We have presented the `AdPreqFr4SL`, which attempts to handle *cost vs. performance* and cope with *concept drift*. Instead of selecting a particular class of BNCs and using it during all the learning process, we propose to use the class of k-DBCs and start with the simple NB by setting $k = 0$. Then, we use simple

control strategies to decide when to do the next move in the spectrum of attribute dependencies (by gradually increasing k) and to start searching for new dependences. As a result, our strategy leads to the scaling up of the model's complexity slowly enough so that the use of more training data will reduce bias at a rate that also reduces variance and consequently the classification error. This bias control leads to the selection of the optimal *class-model* for the current training data (i.e. the optimal k value), thus avoiding *overfitting* or *underfitting* of the current model to the actual data. Since updating the structure is a costly task, we reduce the cost of updating during the whole learning process by first adapting parameters. We adapt the structure only when there is evidence that the performance stops improving in a desirable tempo. The `AdPreqFr4SL` also includes a method for *handling concept drift* based on the `P-Chart`, which has been demonstrated to be efficient for recognizing concept changes. Since NB is one of the most used classifiers in real-world on-line applications and results evidence significant improvements of the NB over time, obvious topic for this line of investigation include the application of the proposed `AdPreqFr4SL` framework to real-world on-line learning systems.

Acknowledgments

Thanks to the financial support given by the FEDER, the Plurianual support attributed to LIACC, project ALES II (POSI/EIA/55340/2004).

References

1. Brumen, B., Golob, I., Jaakkola, H., Welzer, T. and Rozman, I.: Early Assessment of Classification Performance, *Australasian CS Week Frontiers* (2004) 91–96.
2. Castillo, G., Gama, J. and Medas, P.: Adaptation to Drifting Concepts. In *Progress in Artificial Intelligence*, LNCS **2902**, Springer-Verlag (2003) 279–293
3. Castillo, G. and Gama, J.: Bias Management of Bayesian Network Classifiers. In *Proceedings of DS 2005*, LNAI **3735**, Springer-Verlag (2005) 70–83
4. Castillo, G.: Adaptive Learning Algorithms for Bayesian Network Classifiers. PhD. Dissertation, University of Aveiro, (2006).
5. Friedman, N., Geiger, D. and Goldszmidt, M.: Bayesian Network Classifiers. *Machine Learning* **29** (1997) 131–161.
6. Dawid, A.P.: Statistical theory: The prequential approach, *Journal of the Royal Statistical Society A***147** (1984) 278–292.
7. Gama, J.: Iterative Bayes. In *Theoretical Computer Science*, **292-2** Elsevier Science (2003) 417–430.
8. O'Rourke, J.O.: Computational Geometry in C, Cambridge University Press (1992)
9. Sahami, M.: Learning Limited Dependence Bayesian Classifiers, In *Proceedings of KDD-96*, **10** AAAI Press (1996) 335–338.
10. Sen, P.K.: Estimates of the regression coefficient based on Kendall's tau. In *Journal of the American Statistical Association.* **63** (1968) 1379–1389
11. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Context. Machine Learning **23** (1996) 69–101.

Adaptive Active Classification of Cell Assay Images

Nicolas Cebron and Michael R. Berthold

ALTANA Chair for Bioinformatics and Information Mining
Department of Computer and Information Science
University of Konstanz
Box M 712, 78457 Konstanz, Germany
{cebron, berthold}@inf.uni-konstanz.de

Abstract. Classifying large datasets without any a-priori information poses a problem in many tasks. Especially in the field of bioinformatics, often huge unlabeled datasets have to be explored mostly manually by a biology expert. In this work we consider an application that is motivated by the development of high-throughput microscope screening cameras. These devices are able to produce hundreds of thousands of images per day. We propose a new adaptive active classification scheme which establishes ties between the two opposing concepts of unsupervised clustering of the underlying data and the supervised task of classification. Based on Fuzzy *c*-means clustering and Learning Vector Quantization, the scheme allows for an initial clustering of large datasets and subsequently for the adjustment of the classification based on a small number of carefully chosen examples. Motivated by the concept of active learning, the learner tries to query the most informative examples in the learning process and therefore keeps the costs for supervision at a low level. We compare our approach to Learning Vector Quantization with random selection and Support Vector Machines with Active Learning on several datasets.

1 Introduction

Traditionally, a classifier is built on a given set of labeled training data. This is known as supervised learning, as the classifier gets supervision in the form of labeled instances. This can be very useful in many settings - however, sometimes a large pool of unlabeled data is available and the cost of determining the class label for all these examples is prohibitively high. An example for such a setting may be the categorization of web pages, where we have a small set of labeled webpages and a large set of unlabeled examples.

One traditional technique to make use of unlabeled data is clustering - grouping objects that are similar to each other. It is classically used to reveal the underlying structure of the given data. The most important advantage of this method is that it can be used without any supervision by the user. This technique is known as unsupervised learning.

There are also semi-supervised learning techniques that take advantage of a small pool of labeled examples that help to guide the algorithm; they are still

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 79–90, 2006.
© Springer-Verlag Berlin Heidelberg 2006

influenced by the unlabeled data. Examples for techniques that use a small set of labeled examples in clustering can be found in [15] and [1].

A more recent approach is the concept of active learning [4]. Active learning handles the setting where a large pool of unlabeled samples is available and where we have access to a (usually noiseless) oracle, often a human expert, that can determine the class label of an instance. The examples to query are chosen by the learner with a certain strategy so as to optimize the prediction accuracy while at the same time keeping the number of queries low.

In this work, we consider a more special setting that is based on the classification of cell assay images (see Section 4). In our scenario, a large number of unlabeled images of cell assays are available, whereas we only have a human biology expert who is able to provide us with class labels for each cell image.

As we do not have any labeled instances at the beginning, we introduce a new approach that establishes ties between the opposed methods of unsupervised and supervised learning. First, the dataset is explored to find the groupings (hopefully related to possible clusters of the same class) whereas in the second step, the accuracy of the classifier is optimized by querying "useful" examples.

In Section 2, we recapitulate the concept of active learning. Section 3 describes the Fuzzy c-means algorithm with noise detection and the Learning Vector Quantization algorithm, which formed the foundation for our proposed adaptive active clustering scheme that is described in more detail at the end of this section. A useful application for this scheme – mining of cell assay images – is explained in Section 4. We study the behavior of our algorithm and compare it to other methods in Section 5, before we draw conclusions in Section 6.

2 State of the Art

In many classification tasks it is common that a large pool of unlabeled examples U is available whereas the cost of getting a label for an example is high. The concept of active learning [4] tackles this problem by enabling a learner to pose specific queries, chosen from an unlabeled dataset. In this setting, we assume that we have access to a noiseless oracle that is able to predict the class label of a certain sample. Given an unlabeled dataset U, a labeled dataset L and a set of possible labels C, we can describe an active learner as a tuple (f, q). $f : L \cup U \mapsto C$ is the classifier, trained on the labeled (and sometimes also the unlabeled) data. The query function q makes a decision based on the currently labeled samples, which examples from U should be chosen for labeling. The active learner returns a new classifier f' after each pool query or a fixed number of pool queries.

Many active learning strategies for different kinds of algorithms exist. In [4], a selective sampling is performed according to where the most general and the most specific hypotheses disagree. The hypotheses were implemented using feed-forward neural networks with backpropagation. Active Learning with Support Vector Machines (SVM) has also become very popular. The expensive learning process for the SVM can be reduced by querying examples with a certain

strategy. In [16], the query function chooses the next unlabeled data point closest to the decision hyperplane in the kernel induced space. Support Vector Machines with active learning have been widely used for image retrieval problems [12] [17] or in the drug discovery process [18]. However, they require at least some labeled examples from the start, which usually results in some randomly chosen examples to be queried, which is rather inefficient.

To make use of the underlying distribution of the given unlabeled data, we use an approach that clusters the data. To date, research on approaches that combine clustering and active learning has been sparse.

In [1], a clustering of the dataset is obtained by first exploring the dataset with a *Farthest-First-Traversal* and providing *must-link* and *cannot-link* constraints. In the second *Consolidate*-phase, the initial neighborhoods are stabilized by picking new examples randomly from the dataset and again by providing constraints for a pair of data points.

In [7], an approach for active semi-supervised clustering for image database categorization is investigated. It includes a cost-factor for violating pairwise constraints in the objective function of the Fuzzy c-means algorithm. The active selection of constraints looks for samples at the border of the least well-defined cluster in the current iteration.

However, our approach differs from the others in the way that the data is preclustered before supervision enhances the classification accuracy. Thus, our scheme is able to first explore and later finetune the classification of a large unlabeled dataset in an efficient and accurate way.

3 Active Classification

In this section, we present our new active classification scheme. Starting with a short revision of the Fuzzy c-means algorithm (with noise detection) in 3.1 and the Learning Vector Quantization (LVQ) algorithm in 3.2, we propose a query function in 3.3 and put the pieces together for our adaptive active classification scheme in 3.4.

3.1 Fuzzy c-Means with Noise Detection

The Fuzzy c-means (FCM) algorithm [2] is a well-known unsupervised learning technique that can be used to reveal the underlying structure of the data based on a similarity measure. Fuzzy clustering allows each data point to belong to several clusters, with a degree of membership to each one. We use the extended version from [5] for the added detection of noise.

Let $T = \vec{x}_i,\ i = 1, \ldots, |T|$ be a set of feature vectors for the data items to be clustered, $W = \vec{w}_k,\ k = 1, \ldots, c$ a set of c clusters. V is the matrix with coefficients where $v_{i,k}$ denotes the membership of $\vec{x}_i$ to cluster k. Given a distance function d, the fuzzy c-means algorithm with noise detection iteratively minimizes the following objective function with respect to v and w:

$$J_m = \sum_{i=1}^{|T|} \sum_{k=1}^{c} v_{i,k}^m d(\vec{w}_k, \vec{x}_i)^2 + \delta^2 \sum_{i=1}^{|T|} \left(1 - \sum_{k=1}^{c} v_{i,k}\right)^2 . \quad (1)$$

$m \in (1, \infty)$ is the fuzzification parameter and indicates to what extent the clusters are allowed to overlap each other. The first term corresponds to the normal fuzzy c-means objective function, whereas the second term arises from the noise cluster. δ is the distance from every data point to the auxiliary noise cluster (thus, there are $c + 1$ cluster with the extra cluster serving as the noise cluster). This distance can either be fixed or updated in each iteration according to the average interpoint distances. Objects that are not close to any of the cluster centers $\vec{w}_k$ are therefore detected as having a high membership to the noise cluster. J_m is subject to minimization under the constraint

$$\forall i : 0 \leq \sum_{k=1}^{c} v_{i,k} \leq 1. \tag{2}$$

FCM is often used when there is no a-priori information available and thus can serve as an overview technique. Based on the prototypes obtained from the FCM algorithm, we can classify the dataset by first querying the class label for each cluster prototype and then by assigning each data point the class label of the closest prototype. A common problem is that the cluster structure does not necessarily correspond to the distribution of the classes in the dataset. Therefore, we have to update the cluster prototypes subsequently. As the fuzzy c-means algorithm does not provide any way to do this, we use the Learning Vector Quantization algorithm for this task, which is introduced in the next section.

3.2 Learning Vector Quantization

Learning Vector Quantization [11] is a so-called competitive learning method. The algorithm works as follows: for each training pattern, the nearest prototype is identified and updated. The update depends on the class label of the prototype and the training pattern. If they possess the same class label, the prototype is moved closer to the pattern, otherwise it is moved away. The learning rate ϵ controls the movement of the prototypes. The learning rate is decreased during the learning phase, a technique known as *simulated annealing* [10]. The LVQ algorithm terminates if the prototypes stop changing significantly.

One basic requirement in the LVQ algorithm is that we can provide a class label for each training point $\vec{x}_i$ that is randomly sampled. We assume that the training set is unlabeled – however an expert can provide us with class labels for some selected examples. As we can only label a small set of examples, we need to optimize the queries with a strategy to boost the classification accuracy while keeping the number of queries at a low level. In the next section, we propose a query function that attempts to solve this problem.

3.3 Selection of Patterns to Query

The selection of patterns is of particular importance as it influences the performance of the classification. Assuming access to a noiseless oracle it is vital to gain as much information as possible from the smallest possible number of

examples. We propose a sampling scheme that covers two aspects: exploration and exploitation. This coincides with the ideas proposed in [14] that an active learning scheme should not only refine the decision boundaries but also needs to verify the current hypothesis. The prior data distribution plays an important role. In [3] the authors propose to minimize the expected error of the learner:

$$\int_x E_T \left[(\widehat{y}(x;D) - y(x))^2 | x \right] P(x) dx \tag{3}$$

where E_T denotes the expectation over $P(y|x)$ and $\widehat{y}(x;D)$ the learner's output on input x given training set D. If we act on the assumption that the underlying structure found by the FCM algorithm already inheres an approximate categorization, we can select further examples by querying data points at the classification boundaries. That means we approximately take into account the prior data distribution $P(x)$.

Exploration. In order to have information about the general class label of the cluster itself that represents our current hypothesis, we perform a technique known as *Cluster Mean selection* [6]. Each cluster is split into subclusters. Subsequently, the nearest neighbor of each subcluster prototype is selected for the query procedure. If a subcluster is not homogeneous – meaning, the labels of the subclusters in the current cluster are different – this subcluster is split up again until the labels are homogeneous or we have reached a given recursion depth.

Exploitation. We assume that the most informative data points lie between clusters of different classes that are not well separated from each other. We call these regions *areas of possible confusion*. This coincides with the findings and results in [6] and [13].

To identify the data points that lie on the frontier between two clusters, we propose a new procedure that is easily applicable in the fuzzy setting. Rather than dynamically choosing only one example for the labeling procedure in each step (which would make it too slow), we focus on a selection technique that selects a whole batch of N samples to be labeled. Note that a data item $\vec{x}_i$ is considered as belonging to cluster k if $v_{i,k}$ is the highest among its membership values. If we consider the data points between two clusters, they must have an almost equal membership to both of them. The selection is performed in two steps: first, all data points are ranked according to their memberships to cluster prototypes with different classes. Then, the most diverse examples are chosen from this pool of examples. The ranking is based on the fuzzy memberships and can be expressed for each data point $\vec{x}_i$ as follows:

$$\text{Rank}(\vec{x}_i) = 1 - (\min |v_{i,k} - v_{i,l}|) \quad \forall k, l = 1, \ldots, c \ \wedge k \neq l \tag{4}$$

Note that we also take into account the class label of each cluster. Only if the clusters correspond to different classes, the rank is computed. After all data points have been ranked, we can select a subset with high ranks to perform the

next step: diversity selection. This prevents the active clustering scheme from choosing points that are too close to each other (and therefore are altogether not that interesting). We refer to the *farthest-first-traversal* [8] usually used in clustering. It selects the most diverse examples by choosing the first point at random and the next points as farthest away from the current set of selected instances. The distance d from a data point x to the set S is defined as $d(S, x) = \min_{y \in S} d(x, y)$, known as the *min-max-distance*.

3.4 Adaptive Active Classification

Our adaptive active classification procedure is based on a combination of the techniques mentioned above. All steps are listed in Algorithm 1. We start to cluster our dataset with the fuzzy c-means algorithm, because we expect dense regions in the feature space to occur which are likely to bear the same class label. Therefore, the fuzzy c-means algorithm should give us a good initialization and prevents us from labeling unnecessary instances in the first querying step. The noise detection in the clustering procedure serves the same purpose: rare data points that represent borderline cases should not be selected, as these noise labels would influence classification in a negative way. Furthermore, these samples would be useless for classification. Note that in this way we have the possibility to present strange cases to the user, which is often also of interest. After a batch of N examples has been selected from within each cluster and at the borders of the clusters, the user interaction takes place: the expert has to label the selected examples. The newly labeled samples are then added to the current set of labeled samples L. After this step, the cluster prototypes can be moved based on the training set L.

Algorithm 1. Adaptive Active Clustering Procedure

```
1: L ← 0
2: Perform the fuzzy c-means algorithm with noise detection (unsupervised).
3: Filter out data points belonging to noise cluster.
4: while Classification accuracy not satisfactory do
5:     Select N training examples within the clusters and at the borders.
6:     Ask the user for the labels of these samples, add them to L.
7:     Move the prototypes according to L (supervised).
8:     Decrease the learning rate ϵ.
9: end while
```

One open question is when to stop the movement of the prototypes. The simulated annealing in the LVQ algorithm will stop the movement after a certain number of iterations. However, an acceptable solution may be found earlier, which is why we propose further stopping criteria:

Validity Measures: Such measures can give us information on the quality of the clustering [19]. We employ the within cluster variation and the between

cluster variation as an indicator. This descriptor can be useful for the initial selection of attributes. Naturally, the significance of this method decreases with the proceeding steps of labeling and adaptation of the cluster prototypes.

Classification Gradient: We can make use of the already labeled examples to compare the previous to the newly obtained results. After the labels of the samples inside and between the clusters have been obtained, the cluster prototypes are moved. The new classification of the dataset is derived by assigning to each data point the class of its closest cluster prototype. By comparing the labels given by the user to the newly obtained labels from the classification, we can calculate the ratio of the number of correctly labeled samples to the number of falsely labeled examples.

Tracking: Another indicator for acceptable classification accuracy is to track the movement of the cluster prototypes. If they stop moving because new examples do not augment the current classification, we can stop the procedure.

Visual Inspection: If the data points are linked to images (as in the setting we describe in Section 4), we can make use of this additional information. Instead of presenting the numerical features, we select the corresponding image of the data tuple that is closest to the cluster prototype. We display the images with the highest membership to the actual cluster and the samples at the boundary between two clusters if they are in different classes. The decision whether the classification accuracy needs improvement can be made by the user based on this visual inspection.

4 Application: Cell Assay Mining

The development of high throughput imaging instruments, e.g. fluorescence microscope cameras, resulted in them becoming a promising tool to study the effect of drug candidates on different cell types. These devices are able to produce hundreds of thousands of images per day.

The goal of the cell assay image mining is to label a few selected cell images by hand and to automatically label the vast majority of the images afterwards. In order to obtain a classification of one image, it is segmented into small subimages, each containing one cell of the original image. The segmentation allows us to consider the cells separately in order to distinguish between different reactions of cells in the same image. When most of the small subimages are classified, a classification of the original image can be made by a majority decision.

Our Cell Assay Image Mining System consists of three major elements: the segmentation module, the feature extraction module, and the classification element. Obviously the number of data points is very large; because we segment thousands of images into small subimages, we reach an order of millions of images. This dataset is an ideal instance for an active learning scheme. In this setting, the oracle is represented by a biology expert who is able to provide a class label for a cell image that is shown to him.

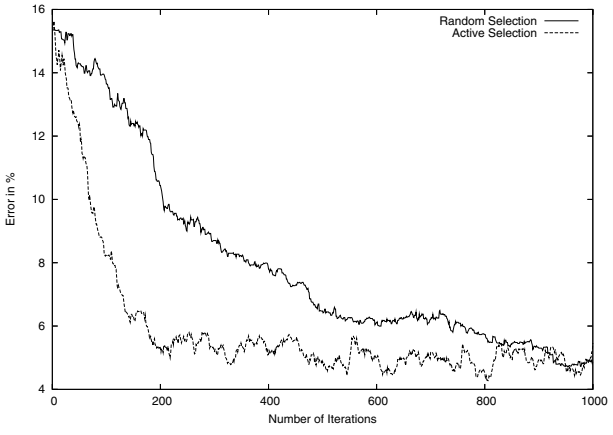


Fig. 1. Error plot using different sampling strategies for 1 pattern per time

The classification of new images is obtained by classifying each individual cell within the given image. Each cell is assigned to a cluster and its corresponding class. The proportion of the distribution of the different classes is the decisive factor for classifying the whole image. If a clear majority decision can be made, the image is not considered further. Borderline cases with equal distributions of classes are sorted into a special container to be assessed manually by the biology expert. It becomes apparent that this approach allows for a rather high fault tolerance, as a human will have no objections to label a few images by hand rather than to risk a misclassification.

5 Experimental Results

To demonstrate the behavior of our adaptive active classification scheme, we first want to show the behavior of our algorithm on artificial data; in the second section we show examples with real world cell image data.

5.1 Artificial Data

The artifical data used in this section is a 2-dimensional dataset consisting of several classes that overlap in the feature space. The class distribution is skewed, taking arbitrary shapes.

In the first setting, we compare our approach to random selection usually used in the LVQ algorithm. As our prototypes are all well initialized, we omit the exploration step (the initial Fuzzy c-means (sub)clustering and labeling) and only focus on the exploitation step of our active classification scheme.

The query function we introduced prevents the LVQ algorithm from choosing instances that are not relevant for classification. The error plot in Figure 1 shows that the active selection leads to a significantly faster convergence of the

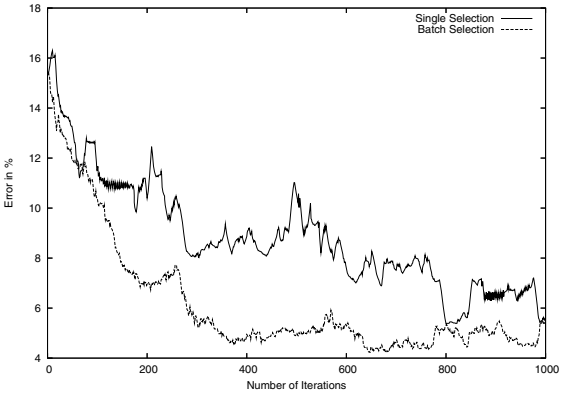


Fig. 2. Classification error of our active classification scheme against active Support Vector Machine on the two-class problem

classification, especially at the first iterations. This is exactly our goal as we want to keep the user interaction at a low level.

Another issue that we want to take a look at is the benefit of batch sampling. One could argue that it is enough to determine the most interesting point at each iteration and then to move the prototypes. We perform a batch sampling that allows a diversity selection to be carried out for performance reasons as well. The benefit of batch sampling is demonstrated in Figure 3, where we plot the error in percent for sampling just one data point at each iteration versus sampling multiple points in each iteration. In fact, the single sampling approach performs much worse than batch selection in this case.

5.2 Cell Assay Image Data

As the cell image data we use is confidential, we show results on a different dataset from the same application area from the NISIS pap-smear competition [9]. The task is to classify pre-stages of cervical cancer in cells before they progress to invasive carcinoma. The data consist of 917 images of Pap-smear cells, classified carefully by cyto-technicians and doctors. Each single cell image is described by 20 numerical features, and the cells fall into 7 classes. A basic data analysis [9] includes linear classification results, in order to provide lower bounds on the acceptable performance of other classifiers. We compared our approach to an approach with a Support Vector Machine with active learning [16]. However, note that the active SVM is initialized differently by choosing random examples from each class. In our setting of cell assay image mining, where we have no labeled instances at the beginning, this step would not be possible, and a random initialization of the SVM would increase the number of queries for the active SVM significantly. Note also, that the performance of the active SVM depends heavily on the chosen kernel function. We used a polynomial kernel, with which the active SVM performed best.

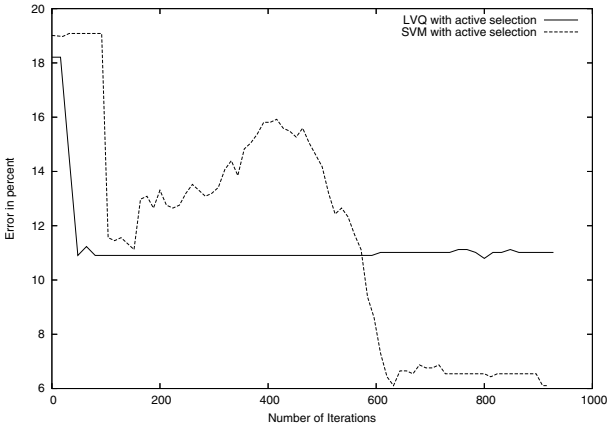


Fig. 3. Single sampling vs. batch sampling (10 examples per batch selection)

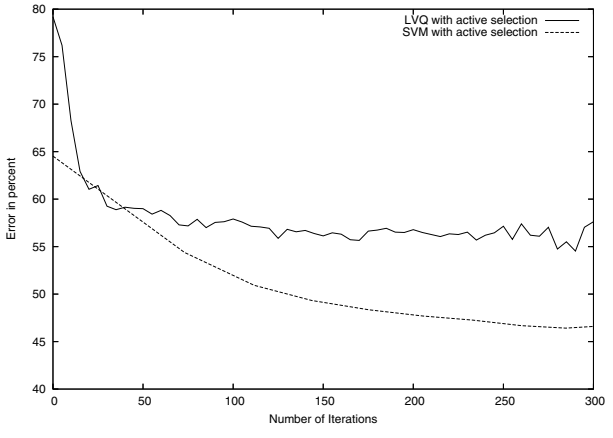


Fig. 4. Classification error of our active classification scheme against active Support Vector Machine on the seven-class problem

The original pap-smear cell dataset with 7 classes can be transformed into a 2-class problem. The results of the comparison between our scheme and the active SVM are shown in Figure 2. The classification error of the linear classifier (trained on 90% of the data) is approximately 10%. As can be seen, both classifiers can reach this performance, the active SVM reaches a classification error of approximately 6% after approximately 600 queries. Our adaptive active classification scheme reaches an error of approximately 11% on the data, however it reaches this performance considerably faster.

On the original 7-class problem, we compared our scheme to active SVM after 300 steps. Since SVMs built binary classifiers, for each class an independent SVM has to be trained against all other classes. Therefore computation for the

optimization of the SVM was not feasible with more steps. Naturally, the batch size of queries for the active SVM is much higher than for our scheme, as we need examples for all classes in each iteration. The results of the comparison can be seen in Figure 4. The classification error of the linear classifier has been given with approximately 40%. Neither the SVM nor our scheme reach this accuracy after 300 queries. The active SVM has an error of approximately 45% whereas our scheme reaches approximately 56%.

From these results we conclude, that our adaptive active classification scheme is a promising approach to tackle the problem of cell assay classification. Its performance is superior to random selection and comparable with a Support Vector Machine with Active Learning on the two-class problem. For the multi-class problem performance is still acceptable but lower than the active SVM. However, the active SVM requires carefully chosen kernels and some pre-labeled examples. Our approach is also computationally more efficient, which is essential for our application where we need to classify tens of millions of cell images.

6 Conclusions

In this work, we have addressed the problem of classifying a large dataset when only a few labeled examples can be provided by the user. We have introduced a new adaptive active classification scheme that starts with the fuzzy c-means algorithm for an initial clustering. The classification of the dataset is obtained by labeling the cluster prototypes and assigning the label of the closest prototype to all data points. We have proposed to move the cluster prototypes, similar to the Learning Vector Quantization (LVQ) method to obtain results closer to the expectation of the user. From the unlabeled pool of instances, new examples are chosen by a query function that makes use of the fuzzy memberships to the cluster prototypes combined with a diversity selection. Based on the labels of the selected examples at the borders between clusters and the labeled examples inside clusters, the prototypes are moved. We have shown that the misclassification rate can be improved more quickly. We have discussed an application in the mining of cell assay images, where the data often inherits the aforementioned properties.

Acknowledgments

This work was supported by the DFG Research Training Group GK-1042 "Explorative Analysis and Visualization of Large Information Spaces".

References

1. S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. *Proceedings of the SIAM International Conference on Data Mining (SDM-2004)*, 2004.
2. J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

3. D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Advances in Neural Information Processing Systems*, 7:705–712, 1995.
4. D. A. Cohn, L. Atlas, and R. E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
5. R. N. Davé. Characterization and detection of noise in clustering. *Pattern Recogn. Lett.*, 12(11):657–664, 1991.
6. B. Gabrys and L. Petrakieva. Combining labelled and unlabelled data in the design of pattern classification systems. *International Journal of Approximate Reasoning*, 2004.
7. N. Grira, M. Crucianu, and N. Boujemaa. Active semi-supervised clustering for image database categorization. *Content-Based Multimedia Indexing*, 2005.
8. Hochbaum and Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.
9. J. Jantzen, J. Norup, G. Dounias, and B. Bjerregaard3. Pap-smear benchmark data for pattern classification, 2006.
10. S. Kirkpatrick, C. D. G. Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
11. T. Kohonen. *Self-Organizing Maps*. Springer Verlag, Heidelberg, 1995.
12. T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, and T. Hopkins. Active learning to recognize multiple types of plankton. *Journal of Machine Learning Research*, pages 589–613, 2005.
13. H. Nguyen and A. Smeulders. Active learning using pre-clustering. *ICML*, 2004.
14. T. Osugi, D. Kun, and S. Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 330–337, 2005.
15. W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *IEEE Transactions on systems, man and cybernetics Part B: Cybernetics*, 27:177–185, 1997.
16. G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. *ICMLProceedings, 17th International Conference on Machine Learning*, pages 839–846, 2000.
17. L. Wang, K. L. Chan, and Z. h. Zhang. Bootstrapping SVM active learning by incorporating unlabelled images for image retrieval. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:629–634, 2003.
18. M. K. Warmuth, G. Raetsch, M. Mathieson, J. Liao, and C. Lemmen. Support vector machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, pages 667–673, 2003.
19. M. Windham. Cluster validity for fuzzy clustering algorithms. *Fuzzy Sets and Systems*, 5:177–185, 1981.

Learning Parameters in Entity Relationship Graphs from Ranking Preferences

Soumen Chakrabarti* and Alekh Agarwal

IIT Bombay
soumen@cse.iitb.ac.in

Abstract. Semi-structured entity-relation (ER) data graphs have diverse node and edge types representing entities (paper, person, company) and relations (wrote, works for). In addition, nodes contain text snippets. Extending from vector-space information retrieval, we wish to automatically learn ranking function for searching such typed graphs. User input is in the form of a partial preference order between pairs of nodes, associated with a query. We present a unified model for ranking in ER graphs, and propose an algorithm to learn the parameters of the model. Experiments with carefully-controlled synthetic data as well as real data (garnered using CiteSeer, DBLP and Google Scholar) show that our algorithm can satisfy training preferences and generalize to test preferences, and estimate meaningful model parameters that represent the relative importance of ER types.

1 Introduction

There is much recent interest [12,14,1] in learning to order entities represented by feature vectors, given a partial order $\prec$ involving some of the entities. The order is defined by a scoring function whose parameters have to be learned. A popular scoring function, suitable for ranking documents in Information Retrieval (IR), is an inner product $\beta' x_i$ between the feature vector x_i representing entity i and an estimated parameter vector β; if $i \prec j$, we want $\beta' x_i \leq \beta' x_j$.

Increasingly, documents are not isolated sequences of words, but are interconnected through a network. This is true not only of the Web, where hyperlinks greatly assist ranking [5,15], but also of entity-relationship (ER) graphs [4,2] and XML data [10] where nodes represent entities with textual attributes and edges represent diverse relations. A sample ER graph is shown in Fig. 1. Activation spreading or Pagerank-like Markovian random walks are often used to score nodes given a query. Typically, a database administrator has to assign and/or tune edge weights, which are used to bias the walks or activation propagation.

In Section 2 we will review a number of efforts to learn some of the parameters of the Markov walk system, most typically via heuristic search [17], quadratic programs [20] or local hill-climbing [7,9].

To our knowledge, no single existing approach covers the scenario we address in Section 3: User preference is provided as $\prec$ (not as absolute score targets

* Contact author.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 91–102, 2006.
© Springer-Verlag Berlin Heidelberg 2006

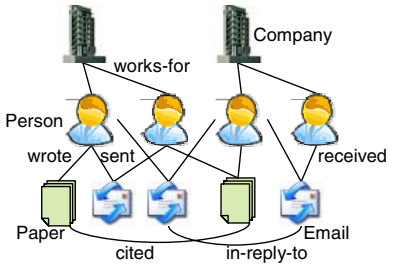


Fig. 1. ER graph with diverse node and edge types

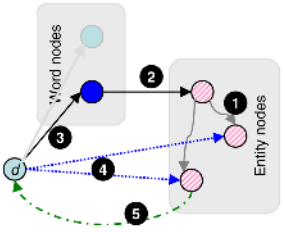


Fig. 2. Typed graph with word nodes

as in some previous work), and ranking is query-specific. We must learn to combine information from feature vectors associated with nodes as well as edge types to output a total order that agrees well with $\prec$ and generalizes to unseen preferences. As a by-product we learn a notion of relative conductivity of different edge types.

In Section 4 we carefully evaluate our proposed algorithm using synthetic data as well as real data from DBLP, CiteSeer and Google Scholar. Unlike some of the earlier work, we give a very detailed account of loss functions, constraints on parameters and model parsimony, the nature of the optimization surface and parameter search techniques. We will release our code and data in the public domain [16].

2 Related Work

Learning to rank feature vectors: Learning to rank items represented by feature vectors from partial orders or point-scale training input (ordinal regression) is well-explored in machine learning [12,14,1]. In RankSVM [14], a slack variable $s_{ij} \geq 0$ is introduced for each constraint $i \prec j$, and the preference is expressed as $\beta' x_j + s_{ij} \geq \beta' x_i + 1$, or, equivalently, $s_{ij} \geq \max\{0, 1 - (\beta' x_j - \beta' x_i)\} = \text{hinge}(1 + \beta' x_i - \beta' x_j)$, where $\text{hinge}(y) = \max\{0, y\}$ is the classic hinge loss. The objective is to minimize $\beta'\beta + B \sum_{i \prec j} s_{ij}$, where B is a magic parameter that trades off the norm of β against the severity of training set errors. Summarizing, RankSVM seeks to minimize $\beta'\beta + B \sum_{i \prec j} \text{hinge}(1 + \beta' x_i - \beta' x_j)$, which can be done using standard quadratic programming tools. However, observe that the training set error is just $\sum_{i \prec j} \text{step}(\beta' x_i - \beta' x_j)$, where $\text{step}(y) = [\![y > 0]\!]$. ($[\![I]\!]$ is 1 if condition I is true and 0 otherwise.) RankSVM upper bounds the training error with $\sum_{i \prec j} \text{hinge}(1 + \beta' x_i - \beta' x_j)$, which is more amenable to quadratic optimization.

Pagerank basics: We review the "random surfer" model of Pagerank [5] briefly. Items are now nodes in a graph $G = (V, E)$, not feature vectors. In the steady state, the random surfer is at node j with probability $p_j = \sum_i p_i p(j|i)$. If we write $p(j|i)$ as a transition or conductance matrix C, the row vector p solves $p = Cp$. C is designed as

$$C(j,i) = \begin{cases} \alpha \frac{[\![(i,j) \in E]\!]}{\text{OutDegree}(i)} + (1-\alpha) r_j, & i \notin \text{leaf}(V) \\ r_j, & \text{otherwise} \end{cases} \quad \text{(UnweightedConductance)}$$

leaf(V) is the set of dead-end nodes without outlinks. The two design variables are α, the probability of walking to a neighbor instead of jumping to a random node; and $r = (r_j)$, the *teleport* or *personalization* vector, which, in ordinary Pagerank, is set uniformly to $(1/n, \ldots, 1/n)$ where $n = |V|$. With r set thus, p depends only on the structure of G and the value of α.

Follow-up work on Pagerank has attempted to modify the teleport vector r to "personalize" the scores heuristically, based on topics [11], words [18,2], or user preferences on graph nodes [13], but the transitions are designed by hand, not learnt from preference data.

Learning link-based ranking: Recently there have been efforts to learn r and even C automatically. Tsoi *et al.* [20] used a quadratic programming approach to optimize only the teleport vector r in (UnweightedConductance), but their formulation had only one kind of edges. Chang *et al.* [7] proposed tuning edge weights for HITS [15] using relevance feedback without a notion of edge types. Nie *et al.* [17] tried local search exhaustively over each edge type. Diligenti *et al.* [9] fit edge weights using back-propagation in a neural network, in case there are known absolute target scores for a few specific pages. In applications, it is easier to collect partial orders between nodes rather than absolute scores. None of the above approaches combine query and per-node feature vectors with link information.

Combining links and text in ranking: Pagerank [5] is precomputed on the entire Web graph and combined with text-based scores at query time in undocumented and proprietary ways. In HITS [15], query keywords drive a heuristic procedure for collecting a limited subgraph of the Web, which is then scored. We know of only a few attempts to combine link and text information for ranking (for the classification task much more is known) in a unified, principled manner. Cohn and Hofmann [8] propose an elegant joint generative model combining text and links, but leave the application to search and ranking unspecified. Silva *et al.* [19] extend Turtle and Croft's approach to IR scoring using Bayesian networks [21] to include link information, but as precomputed scores from standard HITS—the learning is limited to the Bayesian network and there is no learning associated with hyperlink edges.

3 Learning Markov Parameters from Preferences

We are given a directed graph $G = (V, E)$. Edge (i, j) has type $t(i, j)$ belonging to a set of types numbered $1, \ldots, T$. Each type t has an associated importance represented by a weight $\beta(t)$. Thus, edge (i, j) has weight $\beta(t(i, j))$. We will require that our learning algorithm keep these weights positive, to ensure that

the graph topology is not altered by effectively erasing these edges. We will revisit this issue in Section 3.4.

(UnweightedConductance) is modified to use weights as follows. As before, columns are source nodes and rows are destination nodes. Each column adds up to 1. Teleport is handled by adding a dummy node d to the graph, connecting each node $i \in V$ to d and back again, i.e., edges (i, d) and (d, i).

$$C(j,i) = \begin{cases} 0, & i \neq d, j \neq d, i \in \text{leaf}(V) \\ \alpha \frac{\beta(t(i,j))}{\sum_{j'} \beta(t(i,j'))}, & i \neq d, j \neq d, i \notin \text{leaf}(V) \\ 1, & i \neq d, j = d, i \in \text{leaf}(V) \\ 1-\alpha, & i \neq d, j = d, i \notin \text{leaf}(V) \\ r_j & i = d, j \neq d \\ 0, & i = d, j = d \end{cases} \quad \text{(WeightedConductance)}$$

Here "$i \in \text{leaf}(V)$" means i has no outlinks in G. The Pagerank vector $p \in \mathbb{R}^{(|V|+1)\times 1}$ for a given C satisfies $p = C\,p$. In addition, we wish p to satisfy $\prec$, i.e., for each $i \prec j$, we must have $p_i \leq p_j$. Unfortunately, searching over feasible values of both β and p together will introduce problematic quadratic constraints, thanks to the requirement $p = C\,p$.

3.1 Truncating the Recursion

To make the learning problem manageable, we truncate the recursion. In practice, the Pagerank vector is frequently computed via power iteration, by initializing $p = p^0 = \mathbf{1}^{(|V|+1)\times 1}/(|V|+1)$ and iterating $p \leftarrow Cp$ until convergence. The number of iterations needed, which we call the *horizon* H, depends strongly on α but is less sensitive to other aspects of C, and typically grows slowly with the size of the graph. As we shall see, $p = C^H p^0$ is often an excellent approximation, even for modest values of H (10–50). Therefore, the problem reduces to looking for β such that, for each $i \prec j$, $(C^H\,p^0)_i \leq (C^H\,p^0)_j$. Although we describe our learning procedure in terms of a fixed H, in an implementation we need not pick a fixed horizon, but iterate until a specific error tolerance is satisfied.

3.2 Choice of Loss Function

Next we will look at various choices of the loss function. We need to approximate $\sum_{i \prec j} \text{step}((C^H\,p^0)_i - (C^H\,p^0)_j)$. Because $\|p^0\|_1 = 1$ and columns of C add up to 1, $\|p\|_1 = 1$ as well. Therefore $-1 \leq (C^H\,p^0)_i - (C^H\,p^0)_j \leq 1$, and thus picking $\text{loss}(y) = \text{hinge}(1+y)$ is meaningless because we end up just minimizing $\sum_{i \prec j}(1+(C^H\,p^0)_i-(C^H\,p^0)_j)$ where a satisfied constraint contributes a *negative* amount to the sum.

(Note that this is a non-issue for RankSVM because there, $\|\beta\|_2$ can be inflated by the optimizer to prevail over these effects. Moreover, if the training input comprises absolute score targets [9] for nodes, rather than the more realistic partial order preferences, this problem does not arise.)

In optimizing conditional models, it is very common to replace the hinge loss with a "soft" hinge loss of the form $\log(1+e^y)$, which asymptotes to the hinge loss for large $|y|$. Yet another possibility is to directly approximate the step function $\text{step}(y)$ with the logit function $\text{logit}(y) = 1/(1+e^{-y})$. Unfortunately, neither soft hinge nor logit works for us. A detailed study showed that the reason is that they are both positive at $y = 0$ (and small negative values), adding a "noise floor" to the objective even for satisfied preferences, making it very hard for the optimizer to find a reliable gradient. It is very important that $\text{loss}(y)$ is exactly zero for $y \le 0$. After quite some experimentation we picked the Huber loss with window W given by

$$\text{huber}(y) = \begin{cases} 0, & y \le 0 \\ y^2/(2W), & y \in (0, W] \\ y - W/2, & W < y \end{cases}; \quad \text{huber}'(y) = \begin{cases} 0, & y \le 0 \\ y/W, & y \in (0, W] \\ 1, & W < y \end{cases}$$

3.3 Newton Method

All that remains to plug in our formulation into a Newton method is the computation of $(\partial/\partial\beta_t)\sum_{i \prec j} \text{loss}((C^H\, p^0)_i - (C^H\, p^0)_j)$, which is

$$\sum_{i \prec j} \text{loss}'\big((C^H\, p^0)_i - (C^H\, p^0)_j\big)\,\big(\partial(C^H\, p^0)_i/\partial\beta_t - \partial(C^H\, p^0)_j/\partial\beta_t\big)\,.$$

We can compute this easily if we had a $(|V|+1)\times T$ matrix of Pagerank gradients $\frac{\partial}{\partial\beta_t}(C^H\, p^0)_i$ where $i = 1, \ldots, |V|+1$ and $t = 1, \ldots, T$. This matrix can be built up inductively over $h = 0, \ldots, H$ as follows:

$$\frac{\partial}{\partial\beta_t}(C^0 p^0)_i = 0 \quad \text{for all } t \text{ and } i, \tag{1}$$

and for $h = 1, \ldots, H$:

$$\frac{\partial}{\partial\beta_t}(C^h p^0)_i = \sum_j \left[\frac{\partial C(i,j)}{\partial\beta_t}(C^{h-1} p^0)_j + C(i,j)\frac{\partial}{\partial\beta_t}(C^{h-1} p^0)_j\right] \quad \text{(ChainRule)}$$

Finally we compute $\frac{\partial C(i,j)}{\partial\beta_\tau}$ from (WeightedConductance), where the only interesting case is $i \ne d, j \ne d, i \notin \text{leaf}(V)$:

$$\frac{\partial C(i,j)}{\partial\beta_\tau} = \begin{cases} -\alpha\dfrac{\beta(t(i,j))\sum_w [\![\tau = t(i,w)]\!]}{(\sum_w \beta(t(i,w)))^2} & \tau \ne t(i,j) \\ \alpha\dfrac{\sum_w \beta(t(i,w)) - \beta(t(i,j))\sum_w [\![\tau = t(i,w)]\!]}{(\sum_w \beta(t(i,w)))^2}, & \tau = t(i,j) \end{cases} \tag{2}$$

In case we wish to also make α a variable in the optimization, (ChainRule) carries over unchanged, and the nonzero derivatives of conductance are:

$$\frac{\partial C(i,j)}{\partial\alpha} = \begin{cases} \frac{\beta(t(i,j))}{\sum_w \beta(t(i,w))}, & i \ne d, j \ne d, i \notin \text{leaf}(V) \\ -1, & i \ne d, j = d, i \notin \text{leaf}(V) \end{cases}$$

Each iteration of the Newton update takes $O((|V| + |E|)\, H)$ floating point operations. $O(T\,|V|)$ main memory is adequate; the edge list E can be scanned sequentially from disk. We use the BLMVM optimizer [3].

3.4 Keeping the Model Parsimonious

In RankSVM, the model parameters are penalized by a $\beta'\beta$ terms in the objective, which is equivalent to imposing a Gaussian prior with zero mean on each element of β. Zero mean does not make sense for us. In fact, any β_t going to zero may change the topology of G and its Markov properties in serious ways. We can see at least two reasonable choices for penalizing model complexity.

Centered at 1: If we consider Equation (UnweightedConductance) as the most parsimonious model, we should center the β_ts at 1. E.g., we might assess a penalty of $\sum_t (\beta_t - 1)^2$, choosing square loss for computational simplicity. For the reasons above we also need to lower bound each β_t strictly away from zero, which involves yet another magic number that we do not like.

Scale-free floating: There is nothing special about 1 as center; given a solution, we can scale all β_ts by a factor and C in Equation (WeightedConductance) (and therefore p) will remain unchanged. We can therefore lower bound all $\beta_t \geq 1$ without loss of generality, and modify the penalty to try to keep all the β_ts close together without centering any of them: $\sum_{t,t'} (\beta_t - \beta_{t'})^2$. Suppose there are two solution vectors, one a constant multiple of the other. The violations and losses are the same, but the solution with smaller magnitude has lower model penalty, so we will naturally prefer that one.

3.5 Incorporating Queries and Node Features

Much recent work on algorithms to learn ranking functions have used an intrinsic feature vector representation of the objects being ranked, whereas we have, thus far, ignored the objects and considered only the graph that connect them. There was also no notion of a query.

We propose two ways to incorporate node features and queries into our framework. The first implements teleport through word nodes and resembles ObjectRank [2] and the intelligent surfer [18]. The second, more pedestrian approach fits a linear combination of Pagerank-induced and node feature-induced scores.

Teleport through word nodes: We assume each node has a set of associated words, and the query is also a set of words. As shown in Fig. 2, we introduce a node for every word. The dummy node d is connected only to nodes corresponding to query words (edge type "3") and other word nodes are deleted. Each matched word node is connected to all entity nodes where the word occurs (edge type "2"). Entity nodes are interconnected as in the rest of this paper (collectively marked as edge type "1" although there could be more than one type of edges here). d also connects to entities directly (edge type "4")—this sets up a competition

between text match and network prestige. Finally, entity nodes teleport back to d as usual (edge type "5"). Our algorithm has to be invoked on this graph for each query and query-specific preferences.

Linear score combination: The above approach retains a "pure Markovian flavor", but requires a query-time Pagerank computation, which is expensive. Practical implementations are more likely to adopt our second approach.

In this approach we first compute a match function $\mu(q, i) \in \mathbb{R}^+$ between the query and the node features. E.g., each node may be a document, and the query and documents may be represented in vector space and μ may be the commonly-used cosine similarity between q and the text of node i. For a fixed query, we will omit q and use μ_i in place of $\mu(q, i)$. For simplicity we will assume that the text score vector μ has been scaled so that $\|\mu\|_1 = 1$.

We use uniform teleport to compute $C^H p^0$, but integrate signal from μ by writing the score vector as $p = (1-\gamma)C^H p^0 + \gamma\mu$, where $\gamma \in [0, 1]$ is part of the optimization, with

$$\frac{\partial}{\partial\gamma} \sum_{i \prec j} \text{loss}(p_i - p_j) = \sum_{i \prec j} \text{loss}'(p_i - p_j) \left[\mu_i - \mu_j + (C^H p^0)_j - (C^H p^0)_i\right].$$

4 Experiments

Here are the main steps of our evaluation scheme:

1. Get G from real data or a synthetic graph generator.
2. Get $\prec$ from real data and do a test-train split, or, for synthetic generation:
   (a) Compute unweighted Pagerank p_{uw} on G using (UnweightedConductance).
   (b) Assign hidden parameters β (and possibly α and γ), and compute weighted Pagerank p_{w} using the hidden parameters and (WeightedConductance).
   (c) Draw stratified samples (typically 1:1 for us) from agreements and disagreements between scores of node pairs as per p_{uw} and p_{w}. This reflects the reasonable null hypothesis of (UnweightedConductance), and the belief that training data must expose the implausibility of the null hypothesis.
3. Give our algorithm G and $\prec_{\text{train}}$ but not the hidden weights.
4. Our algorithm estimates β^* (and possibly α^* and γ^*).
5. Compute weighted Pagerank p^*_{w} using these estimated parameters.
6. Evaluate what fraction of $\prec_{\text{test}}$ is satisfied by p^*_{w}.

Graphs: We experimented with synthetic and real-life graphs. SynthDBLP, a synthetic citation graph, had author, affiliation and paper nodes (total 21000) connected by "works-for", "wrote" and "cited" edges (total 128592). SynthIMDB, a synthetic graph modeling `http://imdb.com`, had actor, movie, and genre nodes (total 21000) with "acted-in" and "belongs-to" edges (total 97121). We used RMAT [6] to generate the synthetic graphs that satisfy typical properties of social networks. The real citation graph curated from DBLP and CiteSeer had author, paper, and venue (conference/journal) nodes (total 147870) and "cited", "wrote" and "appeared-in" edges (total 1145393).

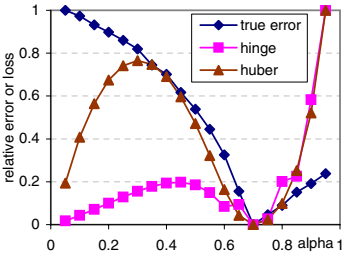


Fig. 3. Training error and losses vs. α ($\alpha_{\text{hidden}} = 0.7$)

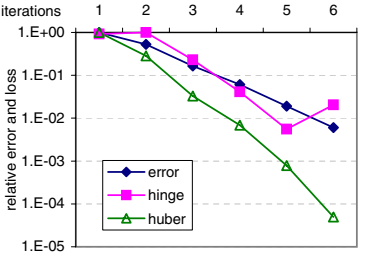


Fig. 4. Training error and losses vs. optimizer iterations

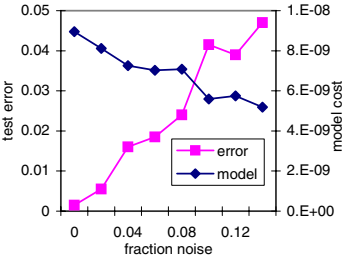


Fig. 5. Test error and model penalty vs. training noise

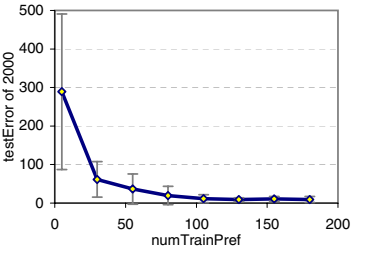


Fig. 6. Test error and its standard deviation (10 runs) vs. $|\prec_{\text{train}}|$

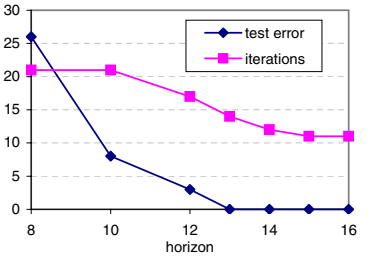


Fig. 7. Effect of H on convergence and test error

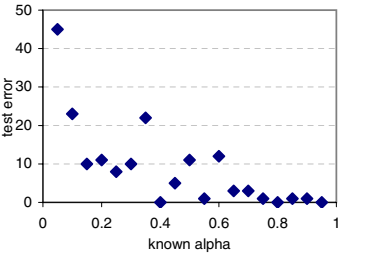


Fig. 8. Test error vs. α

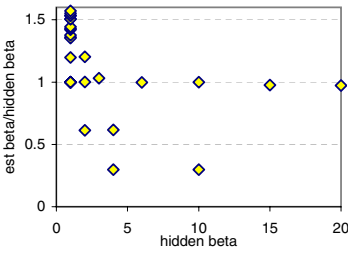


Fig. 9. β estimation accuracy

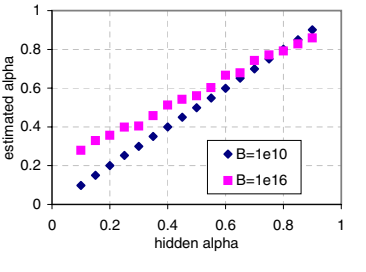


Fig. 10. Estimated vs. hidden α

Edges and weights: Except for experiments involving teleport through word nodes, each edge is made bidirectional, and "hidden" weights are associated with each direction to reflect heuristic values used in the literature [4,2]. Part of our goal is to see if the algorithm can discover these hidden weights given preference data. (Details about data generation can be found in an extended version of this paper [16].)

Error, loss, and convergence: The success of the optimizer depends on how closely our loss approximation tracks training loss. Our general experience is that Huber loss works well for large α (say, larger than 0.4) typically used in Pagerank, but deteriorates at small α. This is not a problem with Huber loss *per se*, because hinge loss (without a margin, defined as $\sum_{i \prec j} \max\{0, p_i - p_j\}$) is an even poorer approximation. Fig. 3 shows training error, hinge loss and Huber loss for variate α. For values of α commonly used in Pagerank algorithms, Huber loss gave reliable convergence (Fig. 4).

Robustness to noisy training data: In real life, our hypothesis that ranking is determined by weighted edges may not hold, and relevance feedback may not even be consistent. Our algorithm seems very robust to random flipping ($i \prec j$ replaced by $j \prec i$) of training pairs (Fig. 5). Even when over 20% of $\prec$ has been corrupted, the error on (clean) test data is less than 6%. It is also reassuring to note that as noise increases, the algorithm cuts back on investments in model complexity (measured as the floating penalty).

Learning rate and validation: To check if the model generalizes well, we generated $\prec_{\text{train}}$ and $\prec_{\text{test}}$ to be *node disjoint*, so that our algorithm cannot, e.g., benefit from discovering transitivity. We increased $\mid \prec_{\text{train}} \mid$ and plotted the test error in Fig. 6. Just a few hundred preference pairs appear adequate to learn a model that generalizes well.

Effect of horizon choice H: Is the truncation of iterations in Section 3.1 reasonable for both objective and gradient? Fig. 7 shows the effects of varying horizon H on the number of iterations to convergence and the error rate (out of 4000 node pairs). As H is increased, the objective and gradient estimates become more accurate (but computationally expensive) and the Newton method converges in fewer iterations. Furthermore, the edge model learnt is more accurate and therefore test error reduces. Even for larger real-life graphs, it appeared that $H > 30$ is always sufficient.

Effect of known α: As α goes to zero, edge weight tuning struggles harder and test error goes up, although, even at $\alpha = 0.05$, test error is less than 5%. Note that in Fig. 8, the algorithm knows α and estimates only the βs.

Edge weight (β) discovery: Fixing G, we varied 2–3 hidden β_ts at a time and ran our algorithm. Recall that all β_ts can be scaled arbitrarily without changing conductance C. In principle, our model penalty should force an automatic scaling

down, but the complex optimization surface can prevent the scaling down once training error is minimized. Therefore we scaled all β_ts by their minimum value. In Fig. 9 we plot the ratio of estimated to hidden β_ts against the hidden value.

Typically the two largest β_ts are estimated very well, but, thanks to our regularization scheme, there is a slight upward pressure on small values and a downward pressure on large values. However, note that the optimization problem is fundamentally degenerate in that, while we start from a specific hidden β, the same Pagerank ordering can be achieved by an infinite number of β vectors, so the deviations below the diagonal hurt neither training nor test error.

Combined α, β search: Fig. 3 shows that the approximate loss surface has local minima. Therefore, a Newton method will need multiple restarts to locate the global optimum. We varied α between 0 and 1, but this was hidden (as was β) from the algorithm, which had to estimate α and β together.

Fig. 10 shows that α is reconstructed with surprising perfection, despite our ignorance of both α and β_ts, (the latter were all initialized to 2, which always led us to the global optimum for any fixed α—this, and the much better quality of reconstructing α, merit a future study). Compared to the fixed α case, more care was needed with B to avoid overfitting.

We performed all our experiments above using synthetic $\prec$ on both synthetic graphs and the real graph culled from DBLP and CiteSeer, and all the results were very similar and consistent.

Integrating queries and text match: We collected queries in the areas of databases and XML (Fig. 11). First, for each query, we pinned down all edge weights except for dummy-to-word edges, which we varied to inspect the rankings obtained. Fig. 12 shows the results. For small dummy-to-word weights, text match is ignored and generic classic papers are listed at the top, whereas at larger weights, the query gets more attention (but citations are still important). Given there are only a handful of query words and about 80000 papers, naturally the dummy-to-word edges need to be quite heavy to have an effect.

Second, we set a hidden weight for dummy-to-word edges and fixed all other edge weights at 1, and tested if our algorithm can discover the hidden weight. The results broadly paralleled our study on other $\beta(t)$s and are omitted. The accuracy was not as good as in Fig. 9. Overestimates like $\beta_{\text{hidden}} = 100$ and $\beta_{\text{estimated}} \approx 20000$ were seen, but training and test errors went down reliably as before. Therefore, the "teleport through word nodes" model works as intended.

Third, we sent the queries to Google Scholar (`http://scholar.google.com`) and sampled the (prefix of the) total order returned to derive $\prec_{\text{train}}$ and $\prec_{\text{test}}$.

```
1: database xml structure index inverted, 2: "data streams" "query processing",
3: database concurrency control deadlock handling, 4: recovery shadow paging,
5: relation nested subquery optimization, 6: transaction serializability,
7: query processing sensor networks, 8: set "similarity join", 9: xml twig join,
10: heterogeneous schema integration "machine learning"
```

Fig. 11. Queries for DBLP and CiteSeer

`transaction serializability`, $\beta(\text{dummy} \rightarrow \text{word}) = 1$
Graph based algorithms for boolean function manipulation (506)
Scheduling algorithms for multiprogramming in a hard real time environment (413)
A method for obtaining digital signatures and public key cryptosystems (312)
Rewrite systems (265)
Tcl and the Tk toolkit (242)
`transaction serializability`, $\beta(\text{dummy} \rightarrow \textit{word}) = 10^6$
On serializability of multidatabase transactions through forced local conflicts (38)
Autonomous transaction execution with epsilon serializability (6)
The serializability of concurrent database updates (104)
Serializability a correctness criterion for global concurrency control in interbase (41)
Using tickets to enforce the serializability of multidatabase transactions (12)

Fig. 12. β(dummy→word) gives a learnable trade-off between word match and citation popularity. Top paper nodes shown with number of citations.

We injected this $\prec$ into the "teleport through word nodes" model with only one variable edge weight, β(dummy → word); there was no known "ground truth". Estimates were between 1 and 2 in 4 of 10 cases, and 21, 127, 172, 509, 650, and 6686 for the others. Training error was typically around 25% but test error was higher, around 35–40%; see comments at the end of this Section.

Finally, to test the "linear score combination" model, we used the IR TFIDF cosine score between the query and the paper titles to obtain μ, and used $\prec$ from Google Scholar. We set a few values of α by hand and estimated γ and β. For small α, G asserts little effect, the Pagerank distribution is very flat, so the optimizer can afford and prefers very small γs. For large α, G provides some valuable signal (absolute error goes down nicely), but it becomes more important to emphasize text: γ rises substantially (Fig. 13).

$\alpha = 0.05$

Q#	$\lvert\prec_{\text{test}}\rvert$	Errors	γ
q1	945	390	0.0272911
q3	771	310	0.0272881
q5	1008	406	0.0209949
q4	993	409	0.0272896

$\alpha = 0.7$

Q#	$\lvert\prec_{\text{test}}\rvert$	Errors	γ
q1	945	326	0.9888147
q3	771	219	0.9364531
q5	1008	382	0.6016729
q4	993	343	0.9833497

Fig. 13. Fitting γ with fixed α

The results involving $\prec$ from Google Scholar are preliminary and come with an important caveat: Google Scholar uses a. paper body text and b. a much larger and different graph compared to our sample of DBLP and CiteSeer, so its ranking function is using information not accessible to us.

5 Conclusion

We have presented models and numerical methods for learning Markov and other parameters for ranking nodes in ER graphs from partial order preferences. The optimization surfaces involved are not always benign, but they must be searched satisfactorily, given the widespread and increasing applicability of such models. We initiate an in-depth treatment of the choice of loss functions, optimization surfaces, search procedures, and parameter settings. In ongoing work we are

investigating text-match models more deeply and seeking to extend the loss framework to become rank-sensitive.

References

1. S. Agarwal, C. Cortes, and R. Herbrich, editors. *Learning to Rank*, NIPS Workshop, 2005.
2. A. Balmin, V. Hristidis, and Y. Papakonstantinou. Authority-based keyword queries in databases using ObjectRank. In *VLDB*, Toronto, 2004.
3. S. J. Benson and J. J. Moré. A limited memory variable metric method for bound constraint minimization. Technical Report ANL/MCS-P909-0901, Argonne National Laboratory, 2001.
4. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*. IEEE, 2002.
5. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW Conference*, 1998.
6. D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In *ICDM*. SIAM, 2004.
7. H. Chang, D. Cohn, and A. McCallum. Creating customized authority lists. In *ICML*, 2000.
8. D. Cohn and T. Hofmann. The missing link — a probabilistic model of document content and hypertext connectivity. In *NIPS*, 2001.
9. M. Diligenti, M. Gori, and M. Maggini. Learning Web page scores by error back-propagation. In *IJCAI*, 2005.
10. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked keyword search over XML documents. In *SIGMOD Conference*, pages 16–27, 2003.
11. T. H. Haveliwala. Topic-sensitive PageRank. In *WWW*, pages 517–526, 2002.
12. R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *International Conference on Artificial Neural Networks*, pages 97–102, 1999.
13. G. Jeh and J. Widom. Scaling personalized web search. In *WWW Conference*, pages 271–279, 2003.
14. T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD Conference*. ACM, 2002.
15. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46(5):604–632, 1999.
16. NetRank project. `http://www.cse.iitb.ac.in/ soumen/doc/netrank`, 2006.
17. Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: Bringing order to Web objects. In *WWW Conference*, pages 567–574, 2005.
18. M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *NIPS 14*, pages 1441–1448, 2002.
19. I. Silva, B. Ribeiro-Neto, P. Calado, E. Moura, and N. Ziviani. Link-based and content-based evidential information in a belief network model. In *SIGIR Conference*, pages 96–103, 2000.
20. A. C. Tsoi, G. Morini, F. Scarselli, M. Hagenbuchner, and M. Maggini. Adaptive ranking of web pages. In *WWW Conference*, pages 356–365, 2003.
21. H. R. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *Transactions on Information Systems*, 9(3):187–222, 1991.

Detecting Fraudulent Personalities in Networks of Online Auctioneers

Duen Horng Chau, Shashank Pandit, and Christos Faloutsos

School of Computer Science, Carnegie Mellon University
{dchau, shashank, christos}@cs.cmu.edu

Abstract. Online auctions have gained immense popularity by creating an accessible environment for exchanging goods at reasonable prices. Not surprisingly, malevolent auction users try to abuse them by cheating others. In this paper we propose a novel method, *2-Level Fraud Spotting* (*2LFS*), to model the techniques that fraudsters typically use to carry out fraudulent activities, and to detect fraudsters preemptively. Our key contributions are: (a) we mine *user level features* (e.g., number of transactions, average price of goods exchanged, etc.) to get an initial belief for spotting fraudsters, (b) we introduce *network level features* which capture the interactions between different users, and (c) we show how to combine both these features using a *Belief Propagation* algorithm over a *Markov Random Field*, and use it to detect suspicious patterns (e.g., unnaturally close-nit groups of people that trade mainly among themselves). Our algorithm scales linearly with the number of graph edges. Moreover, we illustrate the effectiveness of our algorithm on a real dataset collected from a large online auction site.

1 Introduction

Given a set of transactions among online auction users, how do we spot fraudsters? Suppose we want to transact with a user *u*, and we want to know how honest he is. Suppose we also have a lot of historical information (product names, amounts sold for, feedbacks from other users, timestamps, etc.), and that we also have a list of user IDs, who have committed frauds in the past. Currently, users of online auction sites can view the past feedbacks of a user *u*, which may very well be fabricated. How can we include the vast amount of historical information about the user and his trading partners, to spot fraud more effectively? In this paper we present *2LFS*, the first systematic approach to attack auction fraud.

Online auctions have gained immense popularity. For example, eBay, the world's largest auction site, had over 192.9 million registered users at the end of Q1 2006, a 31% increase over the previous year [4]. Unfortunately, auction frauds happen, and they are by far the most serious problems that auction sites face today. In 2005, the Internet Crime Complaint Center (IC3) received 231,493 complaints, 62.7% of which were auction frauds. 41% of the victims reported monetary loss with an average loss of $385 [7]. In some elaborate fraud schemes, the total incurred loss was in the order of millions [10].

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 103 – 114, 2006.
© Springer-Verlag Berlin Heidelberg 2006

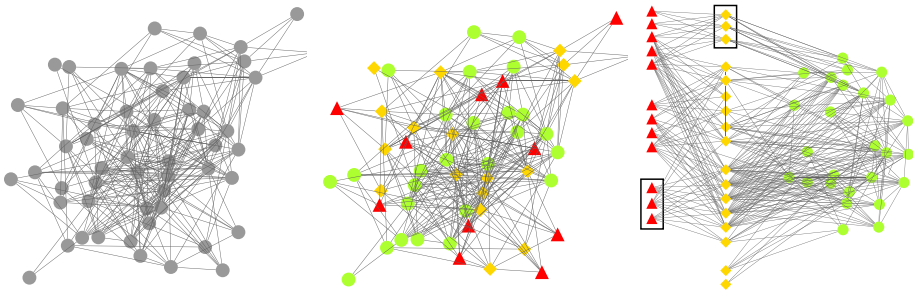


Fig. 1. *2LFS* in action: (a) given graph (b) after labeling by *2LFS*: fraud (red triangles), honest (green circles), "accomplices" (yellow diamonds) (c) after manual rearrangement, to highlight the "bipartite cores". The nodes in the two black rectangles are confirmed fraudsters.

The goal of our work is to treat the auction fraud problem systematically, using data mining and machine learning techniques to spot unnatural patterns in auctions. We propose the *2LFS* algorithm, and illustrate its effectiveness on real, public data from a large auction site. Figure 1(a) illustrates a small graph from the large auction site, in which it is difficult to spot any suspicious patterns. The result of labeling by *2LFS* is shown in Figure 1(b). Fraudsters are the red triangles and honest users are the green circles. The yellow diamonds correspond to *accomplices*, which we will discuss in detail later in the paper. Figure 1(c) shows the same graph after manual rearrangement so that nodes with the same label are grouped together. Now we can clearly observe the existence of a *bipartite core* between the fraudsters and accomplices. As we will explain later, such bipartite cores are a tell-tale sign of a popular fraud scheme. In fact, the nodes in the two rectangles in Figure 1(c) are confirmed fraudsters, who have received many negative feedbacks from buyers who had paid for items that never got delivered.

The rest of the paper is organized as follows. Section 2 provides an overview of related work. Section 3 describes the auction fraud detection problem. Section 4 describes in detail the *2LFS* algorithm. Section 5 provides empirical evidence for the effectiveness, robustness and scalability of our method. Section 6 discusses some observations on how easily we can generalize our method to other fraud detection problems. Finally, we present a brief summary of our results in Section 7 with pointers for future work.

2 Related Work

To the best of our knowledge, this is the first work that uses a systematic approach to analyze and detect electronic auction frauds. We survey earlier attempts to detect such frauds, as well as literature related to trust propagation.

Auction Frauds and Reputation Systems. Reputation systems are extensively used by electronic auctions to prevent frauds. Although helpful, these systems are very simple and can be easily foiled. To study the effectiveness of today's reputation systems,

Melnik et al. [9] and Resnick et al. [14] conducted empirical studies which showed that selling prices of goods are positively affected by the seller's reputation. In an overview, Resnick et al. [13] summarized that today's reputation systems face many challenges which include the difficulty to elicit honest feedback and to show faithful representations of users' reputation. Common-sense approaches to avoid frauds can be easily found on Web sites [5] and in news articles [15]. However, they require people to invest substantial amount of time and to constantly maintain a high level of vigilance, something that the average person cannot afford to do. Some researchers [3] have categorized auction fraud into different types, but they have not suggested any formalized methods to deal with them.

Trust and Authority Propagation. Authority propagation, an area highly related to fraud detection, has been studied by milestone papers and systems, and specifically by PageRank [1] and HITS [8] which treat a Web page as "important" if other important pages point to it, thus propagating the importance of pages over the Web graph. However, none of them explicitly focus on fraud detection. Trust propagation was used by TrustRank [6] to detect Web spam, and their goal was to distinguish between "good" sites and "bad" sites (like phishers, adult-content sites, etc). Also related is the work by Neville et al. [11, 12], where the goal is to aggregate features from neighboring nodes, to do classification, in movie and stock databases.

None of the above techniques focuses on a systematic way to do online auction fraud detection, which is the focus of our work.

3 Problem Description

We define our *Auction Fraud Detection Problem* as follows: **given** (a) a user of interest u (b) the historical information of user u, as well as of many other users and (c) the fact that some of them are known fraudsters, **find** whether user u is a potential fraudster.

We focus on describing the setup of eBay as other auction sites work similarly. A new user begins by registering an ID (also called "handle") with the site. The user may then buy or sell items through bidding and auctioning. All auctions are time-stamped and detailed information about auctions occurring in the last six months is usually available on the site. After a transaction, the site allows the buyer and the seller to rate each other on a scale of positive, neutral and negative (1, 0, -1) and leave a brief comment (e.g., "Great buyer! Prompt payment."). These ratings are added up to form the feedback score of a user. Other users can see the score of a given user before they choose to transact with him. The key idea of the feedback system is to provide an estimate of trustworthiness for each user, that future dealers can consult.

The most prevalent auction fraud is the non-delivery fraud, where the fraudster receives a payment for an item but never delivers it. To be able to commit such a fraud, fraudsters try to devise methods to trick the reputation system and boost their feedback score (we will see how this is done later in the paper.) Other types of frauds include selling faulty, counterfeit, or stolen goods.

4 2-Level Fraud Spotting (*2LFS*) Algorithm

We now present the *2LFS* algorithm, which tackles the fraud detection problem in two steps: (1) it examines *user level features*, i.e., information intrinsic to individual users (e.g., "age" of the user, the number and prices of items sold/bought, the burstiness of the transaction times, etc.), and (2) it examines *network level features* to detect suspicious patterns in the network of transactions between users.

4.1 User Level Features

Auction sites keep records of their users. For each user, we can divide the stored information into two parts, *profile* and *past transactions*. To determine a set of user level features that distinguish fraudsters from honest users, we begin by learning from frauds that were widely publicized in newspaper articles and examining the involved fraudsters. Our observations indicate that fraudsters tend to be short-lived, they exhibit bursty trading patterns (many, fake sales, on a single day) and a bi-modal distribution of prices (cheap items to real, honest users; fictitious, expensive items to their alter-egos).

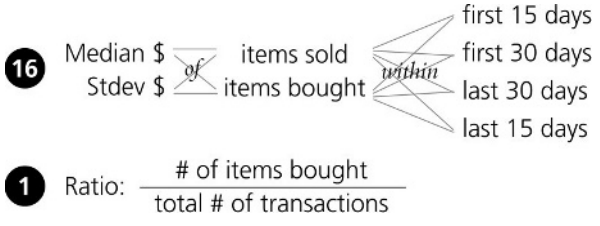


Fig. 2. 17 user level features

We believe that the trends (medians) and fluctuations (standard deviations) in the prices of items traded over time (first 15 days, first 30 days, etc) are the most important features that we should use for classification, as they have direct relevance to fraudsters' investments, costs and profits. The final set of 17 features is summarized in Figure 2. For example, one of the features is *the standard deviation of prices of items sold within the first 15 days since the user registered*. These features were previously evaluated to achieve a precision of 82% and a recall of 83% on some real eBay test data [2].

The feature values can be extracted from the profiles and transaction history of users, available from the Web. The class labels (fraud/honest) are derived manually, by inspecting users with many negative feedback scores. We train a decision tree with the C5.0 classification system, and use this tree to classify other user nodes. These class labels are then fed into the network level detection algorithm described in the next section.

4.2 Network Level Features

Transactions between users can be modeled as a graph, with a node for each user and an edge for all the transactions between two users. As is the case with hyperlinks on

the Web, an edge between two nodes can be assigned a definite semantics, and can be used to propagate properties from one node to its neighbors. For instance, an edge between two nodes can be interpreted as an indication of similarity in their behavior, since honest users will interact more often with other honest users, while fraudsters will interact in small cliques of their own. This semantics is very similar in spirit to the one used by TrustRank[6]. However, preliminary experiments with our dataset, as described in Section 5, suggest that fraudsters do not directly interact with other fraudsters, as this could cause them to suffer extensive "loss" relatively easily – suppose one of the fraudulent account involved in a fraud is exposed, the auction site may easily identify and void other fraudulent accounts in the clique, which would destroy the "infrastructure" that the fraudster had invested in for carrying out the fraud. To carry out another fraud, the fraudster will have to re-invest efforts in building a new clique.

A bit of manual inspection of the data unveiled an alternate way in which fraudsters behave to build a reusable infrastructure for their fraudulent activities. They create several identities and arbitrarily split them into two categories – *fraud* and *accomplice*. The fraud identities are eventually used to carry out the actual fraud, while the accomplices exist only to help the fraudsters by boosting their feedback rating. Accomplices themselves behave like perfectly legitimate users and interact with other honest users to achieve high feedback ratings. On the other hand, they also interact with the fraud identities to form near-bipartite cores, which helps the fraud identities gain high feedback ratings. Once a fraud is committed, the fraud identities get voided by the auction site, but the accomplices manage to beat contemporary fraud detection schemes (owing to their interactions with honest users) and linger around for reuse by new fraudsters. Through *2LFS*, we propose a systematic way to model the network level interactions between users and identify suspicious graph patterns. In a nutshell, each node is given three scores (fraud-, accomplice-, honest-scores), and we update these scores to be in harmony with the neighbors' scores.

The Markov Random Field Model. Markov Random Fields (MRFs) are a class of graphical models particularly suited for solving inference problems with uncertainty in observed data. The data is modeled as a graph with two types of nodes – *hidden* and *observed*. Observed nodes correspond to values that are actually observed in the data. For each observed node, there is a hidden node which represents the true state underlying the observed value. The state of a hidden node depends on the value of its corresponding observed node as well as the states of its neighboring hidden nodes. These dependencies are captured via an *edge compatibility function* $\psi(\sigma,\sigma')$ and a *node compatibility function* $\phi(\sigma,\omega)$. $\psi(\sigma,\sigma')$ gives the probability of a hidden node being in state σ' given that it has a neighboring hidden node in state σ. $\phi(\sigma,\omega)$ gives the probability of a node being in state σ given that its corresponding observation was ω.

We model the auction users and their mutual transactions as a MRF. We create a hidden node for each user, which can be in any of three states – fraud, accomplice, and honest. Let us denote this set of possible states by *S*. A transaction between two users is represented by an edge between their corresponding hidden nodes. With each hidden node *n*, we associate a belief vector $\mathbf{b}_n$, such that $\mathbf{b}_n(\sigma)$ equals the probability of

node *n* being in state σ (which we call the *belief* of node *n* in state σ). Further, each hidden node is also associated with an observed node, which corresponds to our initial (and possibly noisy) observation of its state.

	Fraud	**Accomplice**	**Honest**
Fraud	ε_p	$1-2\varepsilon_p$	ε_p
Accomplice	0.5	$2\varepsilon_p$	$0.5-2\varepsilon_p$
Honest	ε_p	$(1-2\varepsilon_p)/2$	$(1-2\varepsilon_p)/2$

Fig. 3. The Propagation Matrix for an edge. Entry (*i*, *j*) gives the conditional probability that the destination node is at state *j*, when the source node is at state *i*.

	Fraud	**Honest**
Fraud	$1-\varepsilon_o$	ε_o
Accomplice	0	0
Honest	ε_o	$1-\varepsilon_o$

Fig. 4. The Observation Matrix for an edge. Entry (*i*, *j*) gives the *observed* probability that the destination node is at state *j*, when the source node is at state *i*.

To completely define the MRF, we need to instantiate the compatibility functions ψ and ϕ. For now, let us assume that we do not have an initial observation about the states of any of the nodes, and choose ϕ such that $\phi(\sigma,\omega)=1/|S|, \forall\sigma,\omega$. The edge compatibility function can be viewed as a matrix (which we call the *Propagation Matrix*) of dimension $|S|\times|S|$. Figure 3 shows a sample instantiation of the propagation matrix based on the following intuition: a fraudster heavily links to accomplices but not to other fraudsters; an accomplice links to both fraudsters and honest nodes, with a higher affinity for fraudsters; a honest node links to other honest nodes as well as accomplices (since an accomplice effectively appears to be honest to the innocent user.) In our experiments, we set ε_p to 0.05. We would like to set ε_p to zero, but this would create numerical problems with multiplications. Thus we set it to a small value, to denote the fact that it is highly unlikely that a fraudster will have a transaction with another fraudster. Ideally, we would "learn" the value of ε_p, as well as the form of the propagation matrix itself, if we had a large training set.

The Belief Propagation Algorithm. The Belief Propagation (BP) algorithm has been successfully applied to a variety of disciplines (bayesian networks, MRFs, error-correcting codes, etc.) In all of its applications, BP takes as input some form of a network of nodes, each of which can be in a finite number of states. Some encoding of how the state of a node influences its neighbors is also known beforehand. The BP algorithm then infers the posterior state probabilities of all nodes in the network given the observed states of some of the network nodes. We refer the reader to [16] for an excellent discussion on the BP algorithm and its generalizations to various problems.

Here, we present the version of BP suitable for MRFs. The algorithm functions via iterative *message passing* between the different nodes in the network. Let $\mathbf{m}_{ij}$ denote the message that *i* passes to *j*. The message $\mathbf{m}_{ij}$ is a vector with 3 values (fraud-, accomplice- and honest-score), and it represents *i*'s opinion about the belief of *j*. At every iteration, each node *i* computes its belief based on messages received from its

neighbors, and uses the propagation matrix to transform its belief into messages for its neighbors. Mathematically,

$$\mathbf{m}_{ij}(\sigma) \leftarrow \sum_{\sigma'} \psi(\sigma', \sigma) \prod_{n \in N(i) \setminus j} \mathbf{m}_{ni}(\sigma); \qquad \mathbf{b}_i(\sigma) = k \prod_{j \in N(i)} \mathbf{m}_{ji}(\sigma) \tag{1}$$

where k is a normalization constant to make the beliefs sum up to 1. Initially, a suitable prior on the beliefs of the nodes is assumed. The algorithm then proceeds by iteratively passing messages between nodes based on previous beliefs, and then updating beliefs based on the passed messages. The purpose of iteration is to reach a fixed point (equilibrium), that is, status-assignments to nodes that are as compatible with their neighbors as possible. The iteration is stopped as soon as the beliefs converge, or a maximum limit for the number of iterations is exceeded. Theoretically, convergence is not guaranteed, although in practice, BP has been found to converge quickly to reasonably accurate solutions.

4.3 Merging the Two Levels – *2LFS*

We now present the *2LFS* algorithm, which combines the user level features (Section 4.1) with the network level features (Section 4.2) to detect suspicious patterns in a graph of transactions between online auction users.

We treat the user level features as noisy observations of the states of users, and use them to instantiate the observed values of nodes in the MRF model for the network level features. We believe that such a combination would yield the following benefits: (a) suitable prior knowledge will help the belief propagation to converge to a more accurate solution in less time, and (b) incorrect inference at the user level can be corrected by the network level propagation of features.

To combine these observations with the belief propagation, we need to modify the previously stated instantiation of the node compatibility function ϕ. Recall that $\phi(\sigma, \omega)$ gives the probability of a hidden node being in state σ given that the corresponding observation was ω. Let the domain of observed values be Ω. Then, the function ϕ can be encoded as a matrix (which we call the *Observation Matrix*) of dimension $|S| \times |\Omega|$. In our case, the user level features classify users into only two categories – fraud and honest. A sample instantiation of ϕ is shown in Figure 4. ε_o can be interpreted as the uncertainty in observation at the user level. In our experiments, we set $\varepsilon_o = 0.2$.

Let ω_i denote the observed value for node i. To incorporate the effect of the observed values, the update rules in Equation 1 can be extended as follows:

$$\mathbf{m}_{ij}(\sigma) \leftarrow \sum_{\sigma'} \phi(\sigma, \omega_i) \psi(\sigma', \sigma) \prod_{n \in N(i) \setminus j} \mathbf{m}_{ni}(\sigma); \quad \mathbf{b}_i(\sigma) = k \phi(\sigma, \omega_i) \prod_{j \in N(i)} \mathbf{m}_{ji}(\sigma) \tag{2}$$

These equations together constitute the *2LFS* algorithm. The pseudo code for the same is provided in Figure 5.

```
Input:  A: Adjacency matrix, N: Number of nodes, O: Initial observations
        S: Number of states, ψ: Propagation matrix, φ: Observation matrix
Output: B: Belief matrix

NETWORK-LFS()
 1  B = NEWMATRIX(N, S); M = NEWMATRIX(N, N)
 2  for i in 1 to N
 3    M[i][i] = message for observation O[i] /* Initialize priors from user level observations */
 4  while (not converged)
 5   for i in 1 to N
 6     m = NEWMESSAGE() /* Create an empty message */
 7     for k in NEIGBORS(i)
 8       m = MULTIPLYMESSAGE(m, M[k][i]) /* Accumulate messages from all neighbors */
 9     for j in NEIGHBORS(i)
10       m_j = DIVIDEMESSAGE(m, M[j][i]) /* ignore message sent by the neighbor itself */
11       M[i][j] = PROPAGATEMESSAGE(m_j, ψ) /* propagate message to the neighbor */
12   /* Now compute the normalized beliefs for each node */
13  for i in 1 to N
14    m = NEWMESSAGE() /* Create an empty message */
15    for j in NEIGHBORS(i)
16      m = MULTIPLYMESSAGE(m, M[j][i]) /* Accumulate messages from all neighbors */
17    m = NORMALIZEMESSAGE(m) /* Normalize the beliefs */
18    for s in 1 to S
19      B[i][s] = m[s] /* Store beliefs in the belief matrix */
20  return B
```

Fig. 5. Pseudo code for network-LFS

5 Experiments

Here we describe the experiments we did, on real and synthetic datasets. Our goal was to answer the following questions:

1. *Robustness*: how well does *2LFS* work for fraud detection when the topology of the auction graph deviates from the ideal bipartite core setting?
2. *Effectiveness*: how effective is *2LFS* in focusing our attention on suspicious bipartite cores occurring in real auction data?
3. *Scalability*: how well does the network level belief propagation scale with graph size?

The algorithm was implemented in C++ and the experiments were performed on a desktop running Red Hat Linux 9 on a Intel P4 3.00GHz processor, with 1GB RAM, 25GB disk space, and a 100Mbps internet connection.

Robustness of *2LFS*. Graphs observed in practice will almost never have exact bipartite cores; there will always be some "missing" edges. To successfully identify suspicious

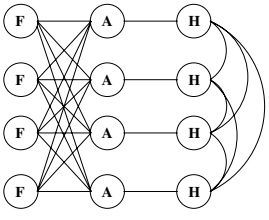


Fig. 6. $G_{3\times4}$ with labels assigned by *2LFS*

nodes in such settings, *2LFS* should be robust in nature and be able to tolerate minor deviations from the ideal scenario. In this section, we describe an experiment we designed to systematically test the robustness of *2LFS*.

We generated synthetic graphs which mimic ideal fraudulent behavior, and then randomly deleted a few edges from them. We started with the graph shown in Figure 6 (called $G_{3\times4}$) which contains 12 nodes – 4 fraud, 4 accomplice and 4 honest. This graph closely agrees with the propagation matrix shown in Figure 3. *2LFS* when run on top of $G_{3\times4}$, converges in 3 iterations and assigns correct labels to all the nodes. Similar results were observed for $G_{3\times x}$ with x varying from 5 to 20.

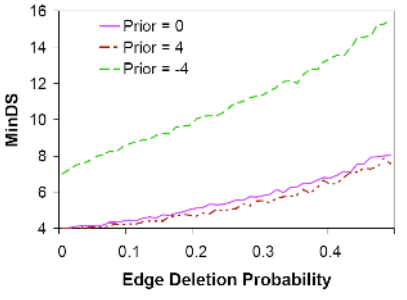


Fig. 7. Min Detection Size vs noise - lower is better: *2LFS* is robust to minor deviations in graph structure, and even to wrong priors

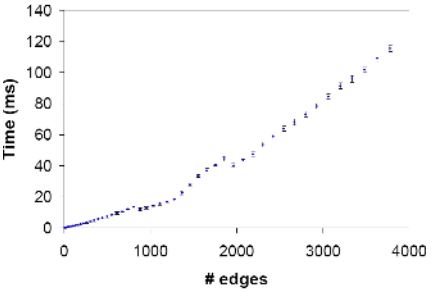


Fig. 8. *2LFS* scales almost linearly with the number of edges

Next, we deleted edges in $G_{3\times x}$ with a fixed probability p (called the *Edge Deletion Probability*). The lowest value of x for which *2LFS* produces the correct labeling is called the *Minimum Detection Size* (MinDS) for the given edge deletion probability. Further, to understand how critically *2LFS* depends on the user level features, we introduced prior observations for some of the nodes in $G_{3\times x}$. Our observations are summarized in Figure 7. Each curve in this figure corresponds to a specific prior value. A prior of 0 means all the nodes were initialized with unbiased priors, a positive prior of z means z nodes were initialized with the correct prior observation, while a negative prior of $-z$ means z nodes were initialized with an incorrect prior. With unbiased priors and edge deletion probabilities below 0.5, the MinDS is 9, which indicates that for large real-world graphs *2LFS* can be expected to robustly tolerate deviations from the ideal $G_{3\times x}$ scenario. Further, minor changes in the prior do not seem to significantly affect the stability of *2LFS*. In case the prior knowledge is correct, performance is improved, while (interestingly) in case the prior knowledge is incorrect, the network level features are able to offset the error in the prior and *2LFS* still converges to the correct solution.

Effectiveness of *2LFS*. To test the effectiveness of *2LFS* we decided to use a dataset crawled from eBay, the world's largest auction site. EBay allows public access to the profiles and feedbacks of (almost) all its users. The feedbacks of a user tell us about

other users with whom he has interacted in the past. The pricing information of the items exchanged is also available only for transactions over the last six months.

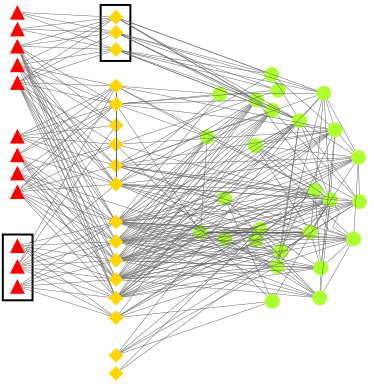


Fig. 9. Labeling of eBay users output by *2LFS*

To propagate the network level features, we rely on a complete and accurate description of the graph. However, user level features (i.e., pricing information) are available only for the last six months. Since the utility of our chosen user level features is evident from [2], we focused on evaluating the effectiveness of the network level features alone, and we set all belief scores to 1/3, for all nodes.

The data was collected by a breadth-first crawl starting from 5 users. The resulting graph had 55 nodes and 620 edges. We then manually observed the feedbacks for each of the 55 users and found six of them to be confirmed fraudsters. Next, we ran *2LFS* over this graph and recorded the labeling assigned to each node in the graph. The entire graph arranged as per the labellings produced by *2LFS* is shown in Figure 9 (same as Figure 1(c)). The nodes labeled as fraud and accomplice form a near bipartite core, one end of which is disconnected from the rest of the graph. As mentioned earlier, the existence of such disconnected bipartite cores is unnatural and suspicious. Moreover, all the six confirmed fraudsters were found to be a part of this core. Thus, *2LFS* clearly succeeds in drawing our attention to suspicious patterns in the graph. Such cores, once identified, can be used to predict which users are likely to commit frauds in the future or are serving to boost the feedback ratings of fraudsters. We believe this aspect of *2LFS* is invaluable in the context of fraud detection.

Scalability of *2LFS*. To test the scalability of our algorithm, we chose to generate synthetic graphs, since we are able to systematically control their sizes and structures. We measured the time taken by *2LFS* to execute over $G_{3\times x}$ for various values of x (averaged over 100 runs.) The results are shown in Figure 8. We observe that the absolute amount of time taken is very small (less than 0.15 seconds for $G_{3\times 3500}$.) Moreover, the running time appears to grow linearly with the number of edges, which indicates that *2LFS* can easily scale to very large real-world graphs.

6 Discussion

We would like to emphasize some important observations. The first is the generality of our method, and the second is its potential for heavy impact on fraudsters.

	Fraud	**Honest**
Fraud	$1-\varepsilon$	ε
Honest	ε'	$1-\varepsilon'$

Fig. 10. Propagation matrix for clique detection ($\varepsilon << \varepsilon'$)

Generality. Thanks to the propagation matrix, *2LFS* is general in applicability. The propagation matrix of Figure 3 can spot bipartite cores. With different instantiations of the propagation matrix, we might be able to spot a broader variety of graph patterns. For example, near cliques could be spotted by the propagation matrix shown in Figure 10.

Making fraud unprofitable. Not only does *2LFS* find confirmed fraudsters, it also spots the accomplices, who help the fraudsters and can themselves commit frauds in the future. Accomplices are valuable to fraudsters, because they take time and effort to build, and they provide a reusable infrastructure for fraudsters to build positive feedbacks quickly. Spotting accomplices can be a hard blow to fraudsters, because accomplices take much more time, money and effort to create and manage. In response, the fraudsters might resort to more sophisticated schemes to hide their evils. However, these schemes will require more effort and cost, thus making fraud increasingly unprofitable for them.

7 Conclusions

We have shown how to use data mining, machine learning and trust propagation methods to address the problem of fraud detection in the complex settings of auction sites. Users and their respective transactions form a rich social network, with much more information than just nodes and links – feedbacks, timestamps, the prices and types of items sold, and more. To handle the complexity of the problem and to exploit useful pieces of information hidden in the social network of auction users, we propose *2LFS*, a novel, two-step algorithm that merges user level and network level information to detect fraudulent users. Our main contributions include:

- The careful extraction of user level features
- The use of belief propagation and MRFs to combine user level and network level features
- Experiments on synthetic and real data, proving the robustness, scalability, and effectiveness of *2LFS*.

Future research directions include the generalization of *2LFS*, so that it can automatically learn the propagation matrix from data, and the inclusion of game theory, to

anticipate and guard against new fraud schemes, in addition to Fraud-Accomplice bipartite cores.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grants No. IIS-0209107 SENSOR-0329549 IIS-0534205. Additional support was provided by by the Pennsylvania Infrastructure Technology Alliance (PITA), by Intel, NTT and Hewlett-Packard. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

References

1. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Seventh international conference on World Wide Web 7 (1998) 107–117
2. D. H. Chau and C. Faloutsos. Fraud detection in electronic auction. In European Web Mining Forum at ECML/PKDD (2005).
3. C. Chua and J. Wareham. Fighting internet auction fraud: An assessment and proposal. In Computer, Vol. 37 no. 10 (2004) 31–37,
4. eBbay 2006 1Q financial results. http://investor.ebay.com/releases.cfm (2006)
5. Federal trade commission: Internet auctions: A guide for buyers and sellers. http://www.ftc.gov/bcp/conline/pubs/online/auctions.htm (2004)
6. Z. Gyongyi, H. G. Molina, and J. Pedersen. Combating web spam with TrustRank. In VLDB (2004) 576–587
7. IC3 2004 internet fraud - crime report. http://www.ifccfbi.gov/strategy/statistics.asp (2005)
8. J. Kleinberg. Authoritative sources in a hyperlinked environment. ACM (JACM), Vol. 46 (1999) 604–632
9. M. Melnik and J. Alm. Does a seller's ecommerce reputation matter? Evidence from eBay auctions. Industrial Economics, Vol. 50 (2002) 337–49
10. Msnbc: Man arrested in huge ebay fraud. http://msnbc.msn.com/id/3078461/ (2003)
11. J. Neville and D. Jensen. Collective classification with relational dependency networks. In 2nd Multi-Relational Data Mining Workshop, 9th ACM SIGKDD (2003) 77–91
12. J. Neville, . Simsek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg. Using relational knowledge discovery to prevent securities fraud. In 11th ACM SIGKDD (2005) 449–458
13. P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. Communications of the ACM, Vol. 43 (2000) 45–48
14. P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood. The value of reputation on eBay: A controlled experiment. (2003)
15. USA Today: How to avoid online auction fraud. http://www.usatoday.com/tech/columnist/2002/05/07/yaukey.htm (2002)
16. J. S. Yedidia,W. T. Freeman, and Y.Weiss. Understanding belief propagation and its generalizations. Exploring artificial intelligence in the new millennium (2003) 239–269

Measuring Constraint-Set Utility for Partitional Clustering Algorithms

Ian Davidson[1], Kiri L. Wagstaff[2], and Sugato Basu[3]

[1] State University of New York, Albany, NY 12222
davidson@cs.albany.edu
[2] Jet Propulsion Laboratory, Pasadena, CA 91109
kiri.wagstaff@jpl.nasa.gov
[3] SRI International, Menlo Park, CA 94025
basu@ai.sri.com

Abstract. Clustering with constraints is an active area of machine learning and data mining research. Previous empirical work has convincingly shown that adding constraints to clustering improves performance, with respect to the true data labels. However, in most of these experiments, results are averaged over different randomly chosen constraint sets, thereby masking interesting properties of individual sets. We demonstrate that constraint sets vary significantly in how useful they are for constrained clustering; some constraint sets can actually decrease algorithm performance. We create two quantitative measures, informativeness and coherence, that can be used to identify useful constraint sets. We show that these measures can also help explain differences in performance for four particular constrained clustering algorithms.

1 Introduction

The last five years have seen extensive work on incorporating instance-level constraints into clustering methods [1,2,3,4,5]. Constraints provide guidance about the desired partition and make it possible for clustering algorithms to increase their performance, sometimes dramatically. Instance-level constraints specify that two items must be placed into the same cluster (must-link, ML) or different clusters (cannot-link, CL). This semi-supervised approach has led to improved performance for several UCI data sets as well as for real-world applications, such as person identification from surveillance camera clips [5], noun phrase coreference resolution and GPS-based map refinement [6], and landscape detection from hyperspectral data [7].

Constraints can be generated from background knowledge about the data set [6,8] or from a subset of the data with known labels [1,2,3,4,5]. Based on the strong positive empirical results that have been reported, the opinion of the community is that constraints help improve clustering performance with respect to accuracy, as measured on the set of extrinsic labels used to generate the constraints. While we might expect that different constraint sets would contribute more or less to improving clustering accuracy, we have found that, surprisingly,

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 115–126, 2006.
© Springer-Verlag Berlin Heidelberg 2006

some constraint sets actually *decrease* clustering performance. We present experimental evidence of this phenomenon in Section 2. We observe that constraints can have ill effects even when they are generated directly from the data labels that are used to evaluate accuracy, so this behavior is not caused by noise or errors in the constraints. Instead, it is a result of the interaction between a given set of constraints and the algorithm being used.

The two major contributions of this work are:

1. The first explicit identification of the adverse effects constraints can have on the clustering process, and
2. The first attempt to characterize constraint set utility to explain clustering performance.

The key question that this work addresses is: *Why do some constraint sets increase clustering accuracy while others have no effect or even decrease accuracy?* We propose two measures, *informativeness* and *coherence*, that capture relevant properties of constraint sets (Section 3). These measures provide insight into the effect a given constraint set has for a specific constrained clustering algorithm. In experiments on several data sets, we find that in general, constraint sets with high informativeness and coherence are most beneficial, and that this trend holds for four different algorithms (Section 4). Finally, we use the CMU Face Images data set [9] to show visual examples of informative and coherent constraint sets.

2 Motivation: Constraints Can Decrease Performance

The operating assumption behind all constrained clustering methods is that the constraints provide information about the true (desired) partition, and that more information will increase the agreement between the output partition and the true partition. Therefore, if the constraints originate from the true partition labels, and they are noise-free, then it should not be possible for them to decrease clustering accuracy. However, as we show in this section, this assumption does not always hold.

The experimental methodology adopted by most previous work in constrained clustering involves generating constraints by repeatedly drawing pairs of data points at random from the labeled subset (which may be the entire data set). If the labels of the points in a pair agree, then an ML constraint is generated; otherwise, a CL constraint is generated. Once the set of constraints has been generated, the constrained clustering algorithm is run several times and the average clustering accuracy is reported. Learning curves are produced by repeating this process for different constraint set sizes, and the typical result is that, on average, when more constraints are provided, clustering accuracy increases [1,2,3,4,5,6,7,8]. However, the focus on characterizing *average* behavior has obscured some interesting and exceptional behavior that results from specific constraint sets. In this work, we will empirically demonstrate such cases and provide insight into the reasons for this behavior.

We begin by examining the behavior of four different constrained clustering algorithms on several standard clustering problems. The two major types of

Table 1. Average performance (Rand Index) of four constrained clustering algorithms, for 1000 trials with 25 randomly selected constraints. The best result for each algorithm/data set combination is in bold.

	Algorithm							
	CKM		**PKM**		**MKM**		**MPKM**	
Data Set	Unconst.	Const.	Unconst.	Const.	Unconst.	Const.	Unconst.	Const.
Glass	69.0	**69.4**	43.4	**68.8**	39.5	**56.6**	39.5	**67.8**
Ionosphere	58.6	**58.7**	58.8	**58.9**	**58.9**	**58.9**	**58.9**	**58.9**
Iris	84.7	**87.8**	84.3	**88.3**	88.0	**93.6**	88.0	**91.8**
Wine	70.2	**70.9**	71.7	**72.0**	**93.3**	91.3	**93.3**	90.6

constrained clustering techniques are (a) direct constraint satisfaction and (b) metric learning. The techniques of the first category try to satisfy the constraints during the clustering algorithm; the latter techniques treat an ML (or CL) constraint as specifying that the two points in the constraint and their surrounding points should be nearby (or well separated) and tries to learn a distance metric to achieve this purpose. We evaluated an example of each kind of algorithm as well as a hybrid approach that uses both techniques:

- COP-KMeans (CKM) performs hard constraint satisfaction [1].
- PC-KMeans (PKM) performs soft constraint satisfaction (permits some constraints to be violated) [4].
- M-KMeans (MKM) performs metric learning from constraints, but does not require that the constraints be satisfied [4].
- MPC-KMeans (MPKM) is a hybrid approach, performing both soft constraint satisfaction and metric learning [4].

Table 1 compares the results (averaged over 1000 trials) for each algorithm in terms of its unconstrained and constrained performance, when provided with 25 randomly selected constraints. We evaluated these algorithms on four UCI data sets [10]: Glass ($n = 214$), Ionosphere ($n = 351$), Iris ($n = 150$), and Wine ($n = 178$). Clustering performance was measured in terms of the Rand Index [11]. The Rand Indices of the unconstrained algorithms differ (e.g., 69.0% for CKM vs. 43.4% for PKM on the Glass data set) because of variations such as different cluster centroid initialization strategies and data pre-processing. In general, as expected and previously reported [1,3,4], average constrained clustering accuracy was equal to or greater than average unconstrained accuracy. The exception is MKM and MPKM's performance on the Wine data set, for which the constraints resulted in a *reduction* in average accuracy.

A careful examination of individual trials reveals that several constraint sets adversely affect clustering performance. Table 2 shows the fraction of these 1000 trials that suffered a drop in clustering accuracy when using constraints, compared to not using constraints. Note that each trial involved the same initialization of the centroids for both the unconstrained and constraint experiments so any change in performance is due to the constraints. We see that, for CKM on all data sets, at least 25% of the constraint sets resulted in a decrease in

Table 2. Fraction of 1000 randomly selected 25-constraint sets that caused a drop in accuracy, compared to an unconstrained run with the same centroid intialization

	Algorithm			
Data Set	**CKM**	**PKM**	**MKM**	**MPKM**
Glass	28%	1%	11%	0%
Ionosphere	26%	77%	0%	77%
Iris	29%	19%	36%	36%
Wine	38%	34%	87%	74%

performance. For the other algorithms, the fraction of negative results ranges up to 77% (for PKM and MPKM) and 87% (for MKM). The high proportion of negative results for MKM and MPKM on the Wine data set help explain why the average results show a decrease in performance (Table 1). These negative results occur frequently for all data sets and algorithms. In fact, only two of the 16 cases presented in Table 2 are completely free of the negative effect (MKM and MPKM with the Ionosphere and Glass data sets respectively). The average performance results tend to mask this effect, since positive gains are often of more magnitude than negative losses. However, for most real applications, we are more interested in performance for the (single) set of available constraints than "average" performance over many sets of constraints.

The possibility of a negative impact from constraints has significant implications for the practice of constrained clustering. First, the assumption that constraints are always helpful (or at least, do no harm) for clustering has been disproven by this empirical evidence. The adverse effects we observe are not restricted to a single data set or constrained clustering algorithm. This underscores the need for a means of characterizing relevant properties of a given constraint set, so that we can understand why it has a positive or negative effect on clustering. Such a characterization can also aid in future studies, so that useful constraints can be selected preferentially and constraints with adverse effects can be avoided. In the next section, we offer two constraint set measures that provide the first steps toward this goal.

3 Characterizing the Utility of Constraint Sets

A major contribution of this work is the introduction of two measures, informativeness and coherence, that quantify important constraint set properties.

- **Informativeness** refers to the amount of information in the constraint set that the algorithm cannot determine on its own. It is determined by the clustering algorithm's objective function (bias) and search preference. For example, in Figure 1(a), an algorithm such as CKM would be biased towards grouping nearby points together and separating distant points, but the specified constraints contradict this bias.
- **Coherence** measures the amount of agreement within the constraints themselves, with respect to a given distance metric. Figure 1(b) shows two

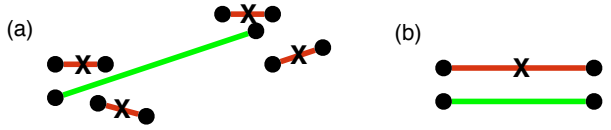


Fig. 1. Simple illustrative examples of (a) constraints with high informativeness for CKM and (b) highly incoherent constraints, given a Euclidean distance metric. Must-link constraints are depicted as solid line segments; cannot-link constraints have an 'X' through them.

constraints (ML and CL) that are very close and parallel. The ML constraint indicates that the distance between the points (and surrounding points) should be small, while the CL constraint implies the opposite. With respect to a Euclidean distance metric, these two constraints are incoherent.

The hypothesis that we investigate in this paper is that *constraint sets with high informativeness and coherence are most likely to provide performance gains.* We also expect that the negative performance effects are caused by highly incoherent constraint sets. First, incoherent constraints (as in Figure 1(b)) can cause metric learning methods (MKM, MPKM) to learn suboptimal global metrics. Also, since these algorithms use the constraints to initialize the cluster centroids, incoherent constraint sets are more likely to lead to a bad cluster initialization, increasing the chance of the clustering algorithm getting stuck in a poor local minimum.

3.1 Quantifying Informativeness

We begin this section with some straightforward but necessary definitions.

Definition 1. ***Partition Specification.*** *For any partition P of a data set D containing n items, a set of constraints C completely* **specifies** *P if it is a set of at most $\binom{n}{2}$ must-link and cannot-link constraints that uniquely defines P.*

Definition 2. ***Incomplete Constraint Set.*** *A set of constraints $\hat{C}$ is* ***incomplete*** *with respect to a data set D if it does not specify a unique partition P of D.*

In practice, most interesting problems will have an incomplete set of constraints, so that there exist multiple partitions that satisfy all constraints. We first introduce an idealized definition of constraint set informativeness.

Definition 3. ***Idealized Informativeness.*** *Let P^* be the partition that globally minimizes the objective function of some algorithm $\mathcal{A}$, in the absence of any constraints. Let C^* specify P^* in the sense given in Definition 1. The informativeness in a given constraint set C is the fraction of constraints in C that are* ***violated*** *by C^*.*

That is, $\{C^* - C\}$ is the set of constraint relationships that $\mathcal{A}$ is unable to correctly determine using its default bias. These constraints are therefore informative with respect to maximizing clustering accuracy. For illustration, consider a data set $\{a, b, c, d, e\}$ with $P^* = \{[a, b], [c, d, e]\}$. Using definition 1, we obtain C^*, which can be compactly represented as $\{ML(a, b), ML(c, d), ML(d, e), CL(a, c)\}$ due to the transitive and entailment properties of ML and CL constraints respectively [1]. If we are given a set of constraints $C_1 = \{ML(a, b), ML(c, d)\}$, then C_1 has an informativeness of 0; each of the constraints was already satisfied by the algorithm's default output P^*. In contrast, $C_2 = \{ML(a, b), ML(b, c)\}$ has an informativeness of 0.5 because $ML(b, c)$ is not in C^* and is therefore new information.

This definition of informativeness cannot be realized in practice, since we do not know P^* prior to clustering. We next present an efficiently computable approximation.

Approximate Measure of Informativeness. Our approximation is based on measuring the number of constraints that the clustering algorithm cannot predict using its default bias. Given a possibly incomplete set of constraints C and an algorithm $\mathcal{A}$, we generate the partition $P_{\mathcal{A}}$ by running $\mathcal{A}$ on the data set without any constraints. We then calculate the fraction of constraints in C that are unsatisfied by $P_{\mathcal{A}}$:

$$\mathcal{I}_{\mathcal{A}}(C) = \frac{1}{|C|}\left[\sum_{c \in C} unsat(c, P_{\mathcal{A}})\right] \tag{1}$$

where $unsat(c, P_{\mathcal{A}})$ is 1 if P does not satisfy c and 0 otherwise. This approach effectively uses the constraints as a hold-out set to test how accurately the algorithm predicts them. Given this equation, we can quantify the informativeness of the constraint sets in Figure 1 for the CKM algorithm as $\mathcal{I}_{CKM}(C_a) = 1.0$ and $\mathcal{I}_{CKM}(C_b) = 0.5$.

3.2 Quantifying Coherence

Coherence is the amount of agreement between the constraints themselves, given a metric $\mathcal{D}$ that specifies the distance between points. It does not require knowledge of the optimal partition P^* and can be computed directly. The coherence of a constraint set is independent of the algorithm used to perform constrained clustering.

One view of an $ML(x, y)$ (or $CL(x, y)$) constraint is that it imposes an attractive (or repulsive) force within the feature space along the direction of a line formed by (x, y), within the vicinity of x and y. Two constraints are incoherent if they exert contradictory forces in the same vicinity. We consider all constraint pairs composed of an ML and a CL constraint (pairs composed of the same constraint type cannot be contradictory). To determine the coherence of two constraints, a and b, we compute the *projected overlap* of each constraint on the other as follows (see Figure 2 for examples).

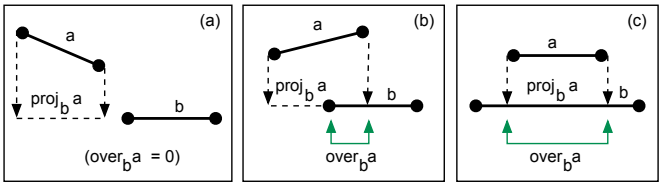


Fig. 2. Three cases of computing the projected overlap between constraints a and b

Let $\overrightarrow{a}$ and $\overrightarrow{b}$ be vectors connecting the points constrained by a and b respectively. Without loss of generality, we use the convention (x_1, x_2) to refer to the points connected by a vector $\overrightarrow{x}$. In the context of Figure 2, x_1 appears to the left of x_2 for all vectors shown. We first project $\overrightarrow{a}$ onto $\overrightarrow{b}$:

$$\overrightarrow{p} = proj_{\overrightarrow{b}} \overrightarrow{a} = \left(|\overrightarrow{a}| \cos\theta\right) \frac{\overrightarrow{b}}{|\overrightarrow{b}|},$$

where θ is the angle between the two vectors. Next, we calculate how much of this projection overlaps with $\overrightarrow{b}$. Since $\overrightarrow{p}$ and $\overrightarrow{b}$ are colinear ($\theta = 0$), we simply compute the distance from b_2 to each of b_1, p_1, and p_2. There are three cases, corresponding to the three examples in Figure 2:

$$over_b a = \begin{cases} 0 & \text{if } d_{b_2,b_1} \leq d_{b_2,p_2}, d_{b_2,b_1} \leq d_{b_2,p_1} \\ d_{b_1,p_2} & \text{if } d_{b_2,p_2} < d_{b_2,b_1}, d_{b_2,p_1} \geq d_{b_2,b_1} \\ d_{p_1,p_2} & \text{if } d_{b_2,p_2} < d_{b_2,b_1}, d_{b_2,p_1} < d_{b_2,b_1} \end{cases} \tag{2}$$

Given this background, we now define coherence, $\mathcal{COH}$, as the fraction of constraint pairs that have zero projected overlap:

$$\mathcal{COH}_{\mathcal{D}}(C) = \frac{\sum_{m \in C_{ML}, c \in C_{CL}} \delta(over_c m = 0 \text{ and } over_m c = 0)}{|C_{ML}||C_{CL}|} \tag{3}$$

We quantify the coherence of the constraint sets in Figure 1 as $\mathcal{COH}(C_a) = 0.0$ (all ML/CL pairs have some overlap) and $\mathcal{COH}(C_b) = 0.0$ (the single constraint pair completely overlaps).

Our measure of coherence is applicable to any space where vector projection is defined. The preceding examples and the experimental results presented later all make use of a Euclidean distance metric, since the four algorithms we evaluate use either Euclidean distance or a close variant, such as a generalized Mahalanobis (weighted Euclidean) distance metric.

4 Experimental Results

In this section, we present three important results. First, we analyze the relationship between the proposed measures (informativeness and coherence) and

Table 3. Average measures of informativeness ($\mathcal{I}$) and coherence ($\mathcal{COH}$) of 5000 randomly generated 3-constraint sets. Compare with Table 1.

	Algorithm				
Data Set	$\mathcal{I}_{CKM}$	$\mathcal{I}_{PKM}$	$\mathcal{I}_{MKM}$	$\mathcal{I}_{MPKM}$	$\mathcal{COH}$
Glass	0.34	0.43	0.50	0.50	0.45
Ionosphere	0.41	0.41	0.42	0.42	0.27
Iris	0.12	0.12	0.11	0.11	0.51
Wine	0.28	0.28	0.06	0.06	0.60

constrained clustering performance. Next, we show the benefits that can be obtained when using these measures to filter constraint sets. Finally, we analyze constraint sets from an image data set and show how informativeness and coherence can provide insights into why clustering performance increases or decreases.

4.1 Impact of Informativeness and Coherence on Clustering Performance

To understand how these constraint set properties affect various algorithms, we performed the following experiment. We randomly generated constraint sets of just three constraints 5000 times. With such a small number of constraints, the possible combinations of informativeness and coherence values is limited, permitting a detailed study. For each data set, we can compare the performance of each algorithm for each possible informativeness/coherence situation.

First, we report the average informativeness and coherence we observed for each algorithm and data set (Table 3). Although Tables 1 and 3 are not directly comparable due to the difference in constraint set sizes, we see an interesting trend. In Table 1, the Glass data set exhibited the largest increases in accuracy when using constraints; we find in Table 3 that the average informativeness for these constraints is also high. However, high informativeness is not sufficient for predicting accuracy improvement: the Ionosphere constraints, although informative, also tend to have very low coherence. Incoherent sets are difficult to completely satisfy, and we see this reflected in the lack of significant improvement when using constraints with this data set. Conversely, the Iris constraints have relatively high coherence but low informativeness, leading to the modest (but positive) average effect on performance for all algorithms. The Wine constraints have a remarkable lack of informativeness for MKM and MPKM, so the incoherence of the data set dominates performance and explains the small decrease in average accuracy.

We have shown that average results can obscure individual behavior. Therefore, we conducted a detailed analysis to better understand the relationships between each measure and performance. Table 4 focuses on constraint sets that are fully coherent, comparing performance between sets with high vs. low informativeness. We find that high informativeness almost always leads to an increase in performance, for all algorithms. The exception is MKM and MPKM on the Wine data set. Table 5 explores the opposite situation, focusing on constraint

Table 4. Average accuracy for fully coherent constraint sets, comparing performance of sets with high ("Inform.") and low ("Uninf.") informativeness

	Algorithm							
	CKM		**PKM**		**MKM**		**MPKM**	
Data Set	Inform.	Uninf.	Inform.	Uninf.	Inform.	Uninf.	Inform.	Uninf.
Glass	**68.9**	67.8	**57.8**	57.1	**58.1**	49.6	**54.9**	54.3
Ionosphere	58.9	58.9	58.8	58.7	58.9	58.9	**93.9**	93.5
Iris	**89.2**	88.1	**88.1**	86.7	**92.9**	89.2	**93.9**	93.5
Wine	71.8	71.8	**72.1**	71.8	92.2	**93.9**	93.5	**93.9**

Table 5. Average accuracy for non-informative constraint sets, comparing performance of coherent ("Coh.") and incoherent ("Incoh.") sets

	Algorithm							
	CKM		**PKM**		**MKM**		**MPKM**	
Data Set	Coh.	Incoh.	Coh.	Incoh.	Coh.	Incoh.	Coh.	Incoh.
Glass	**67.9**	67.4	**57.1**	54.3	**49.6**	49.4	**54.8**	50.2
Ionosphere	58.9	58.9	58.7	58.7	58.9	58.9	58.8	58.8
Iris	**86.7**	85.2	**86.7**	85.2	89.2	89.2	**89.3**	88.8
Wine	71.8	71.8	71.9	71.8	94.0	93.9	**93.5**	93.2

sets that have low informativeness but a variety of coherence values. Incoherence tends to adversely affect performance, particularly for the Glass and Iris data sets. It has less impact on the Ionosphere and Wine data sets.

4.2 Constraint Selection Based on Coherence

We posit that informativeness and coherence can provide guidance in selecting the most useful constraint sets. Returning to the 25-constraint experiments from Section 2, we applied a coarse constraint set selection strategy by removing the 500 least coherent constraint sets and calculating average performance on the remaining 500 sets (Table 6). We find a small but consistent increase in the average accuracy with those sets removed, suggesting that generating or selecting constraint sets with high coherence can provide gains in future constrained clustering experiments. The Iris data set, when analyzed by MPKM, is an exception to this rule. The MPKM results suggest that there are some less-coherent constraint sets that yield very good performance, when both metric learning and constraint satisfaction are used. We plan to investigate this exception more thoroughly in future work.

4.3 Visualizing Informative and Coherent Constraint Sets

We have demonstrated empirically that highly informative and coherent constraint sets lead to improved clustering performance, while incoherent sets can have an adverse effect. In this section, we show examples of constraint sets from

an image data set that permit us to visualize informative and coherent constraint sets.

For these experiments, we used the CMU Face Images data set [9]. We used a subset containing 271 images of human faces with a variety of orientations and expressions. Each image is labeled as Male or Female, and the goal is to identify clusters that correspond with these categories (215 Male and 56 Female images). Each image is approximately 120×120 pixels, yielding a total of 14402 features. We conducted 100 trials, each time generating two randomly selected constraints. Without constraints, the algorithms achieved Rand Indices of: CKM (53.2%), PKM (53.9%), MKM (51.9%) and MPKM (51.9%). When using two randomly selected constraints, the performance ranges were: CKM [53.6%,54.5%], PKM [53.6%, 55.3%], MKM [49.8%,53.7%], MPKM [49.9%, 66.3%]. For this problem, just two constraints can significantly improve the performance of the MKM and MPKM algorithms, suggesting that the constraints are very useful for metric learning.

Since the items in this data set are images, we can directly visualize the constraint sets. Figure 3 shows two constraint sets (one per line) that improved the performance of MPKM from 51.9% to over 65%; both sets have an informativeness and coherence of 1.0. Figure 4 shows two constraint sets (one per line) that either provided no improvement (CKM) or adversely affected performance (PKM, MKM, and MPKM) with respect to the unconstrained performance; both sets have an informativeness and coherence of 0.0.

We see that the beneficial constraint sets have an intuitive interpretation: the must-linked images connect examples with different facial orientations, while the cannot-link constraints are between images with very similar orientations. Because "orientation" is not a feature provided to the algorithm, these constraints are very informative. They encourage the algorithm to create clusters that avoid grouping images simply based on where the bright "face" pixels are located. In contrast, in the constraint sets that have negative effects, the must-linked instances are of different faces in the similar orientation, and the cannot-link constrained instances have different orientation. This biases the constrained clustering algorithms towards clustering faces with the same orientation, which is

Table 6. Clustering performance (Rand Index) when using constraint sets selectively. We report average accuracy over all 1000 25-constraint sets (results copied from Table 1) compared to average accuracy over the 500 most coherent sets. Statistically significant increases at the 95% confidence interval are shown in bold.

	Algorithm							
	CKM		**PKM**		**MKM**		**MPKM**	
Data Set	All	Top 500	All	Top 500	All	Top 500	All	Top 500
Glass	69.4	**70.4**	68.8	**70.6**	56.6	56.6	67.8	**68.4**
Ionosphere	58.6	**59.3**	58.9	58.9	58.8	**59.3**	58.9	58.9
Iris	87.8	**88.3**	88.3	88.3	93.6	**94.5**	**91.8**	91.4
Wine	70.9	**71.5**	72.0	**72.5**	91.3	**93.3**	90.6	**91.1**

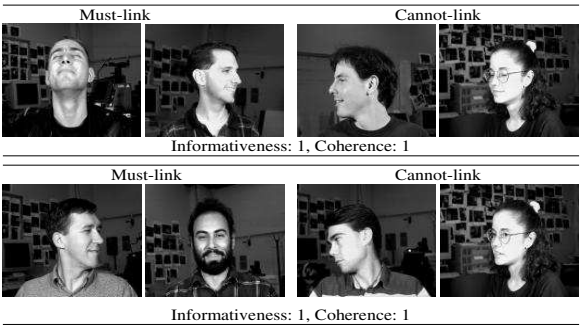


Fig. 3. Examples of beneficial constraint sets (one per line) that significantly improved the performance of MPKM

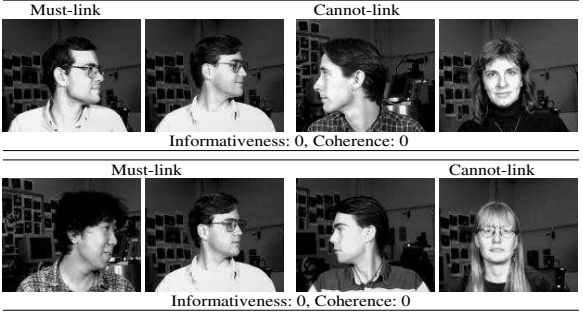


Fig. 4. Examples of constraint sets (one per line) that had no effect or an adverse effect on algorithm performance

not a useful strategy when trying to separate images by gender. Our measures of informativeness and coherence correctly capture this concept by characterizing the likely utility of each set.

5 Conclusions and Future Work

The contributions of this paper are two-fold. First, we have shown the first evidence that constraints can result in a decrease in clustering accuracy. This occurs even with constraints that are completely accurate and noise-free. In experiments with four UCI data sets and four constrained clustering algorithms, we found that the fraction of randomly generated constraint sets that result in a performance drop can range well above 50%. Second, we proposed two constraint set properties, informativeness and coherence, that provide a quantitative basis for explaining why a given constraint set increases or decreases performance. We demonstrated that performance gains are largely attributable to constraint

sets with high informativeness and coherence, while drops in performance are associated with incoherent data sets.

Our experiments with selectively filtering randomly generated constraints to remove sets with low coherence suggest a promising avenue for future work with constrained clustering algorithms. We plan to more fully explore the use of informativeness and coherence to select the most useful constraints for clustering. Ultimately, this research direction could lead to reduced computational effort (since fewer constraint sets are needed to assess performance) and higher average performance on a variety of data sets.

Acknowledgments. We thank the anonymous reviewers for their thoughtful comments. We thank the National Science Foundation (ITR grant #0325329) and DARPA (CALO Project: Contract #NBCHD030010, Order #T310) for partially supporting this work financially. The research described in this paper was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

1. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proceedings of the Eighteenth International Conference on Machine Learning. (2001)
2. Klein, D., Kamvar, S.D., Manning, C.D.: From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: Proceedings of the Nineteenth International Conference on Machine Learning. (2002)
3. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: NIPS 15. (2003)
4. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA (2004)
5. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a Mahalanobis metric from equivalence constraints. Journal of Machine Learning Research **6** (2005)
6. Wagstaff, K.L.: Intelligent Clustering with Instance-Level Constraints. PhD thesis, Cornell University (2002)
7. Lu, Z., Leen, T.K.: Semi-supervised learning with penalized probabilistic clustering. In: Advances in Neural Information Processing Systems 17. (2005)
8. Davidson, I., Ravi, S.S.: Clustering with constraints: Feasibility issues and the k-means algorithm. In: Proceedings of the 2005 SIAM International Conference on Data Mining. (2005)
9. Mitchell, T.: Machine Learning. McGraw Hill, New York, NY (1997)
10. Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998)
11. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association **66**(366) (1971)

Discovery of Interesting Regions in Spatial Data Sets Using Supervised Clustering

Christoph F. Eick, Banafsheh Vaezian, Dan Jiang, and Jing Wang

Department of Computer Science, University of Houston
Houston, Texas 77204-3010, U.S.A
{ceick, bvaezian, djiang, jwang3}@cs.uh.edu

Abstract. The discovery of interesting regions in spatial datasets is an important data mining task. In particular, we are interested in identifying disjoint, contiguous regions that are unusual with respect to the distribution of a given class; i.e. a region that contains an unusually low or high number of instances of a particular class. This paper centers on the discussion of techniques, methodologies, and algorithms to discover such regions. A measure of interestingness and a supervised clustering framework are introduced for this purpose. Moreover, three supervised clustering algorithms are proposed in the paper: an agglomerative hierarchical supervised clustering named SCAH, an agglomerative, grid-based clustering method named SCHG, and lastly an algorithm named SCMRG which searches a multi-resolution grid structure top down for interesting regions. Finally, experimental results of applying the proposed framework and algorithms to the problem of identifying hotspots in spatial datasets are discussed.

1 Introduction

Because of advances in database technologies, data collection techniques, and data gathering devices the amount of spatial data has been growing tremendously in recent years. The goal of spatial data mining is to automate the extraction of interesting and useful patterns that are not explicitly represented in spatial datasets.

This paper centers on discovering interesting regions in spatial datasets; in particular, on identifying disjoint, contiguous regions that are unusual with respect to the distribution of a given class, i.e. a region that contains an unusually low or high number of instances of a particular class. Methodologies, techniques, and algorithms are proposed for this purpose. Challenges that this task faces include the capability to find regions of arbitrary shape and at arbitrary levels of resolution, the definition of suitable parameterized measures of interestingness to instruct discovery algorithms what they are supposed to be looking for, and the need to reduce computational complexity due to the large size of most spatial datasets.

The paper assumes that datasets contain classified examples, and treats region discovery as a clustering problem in which clusters have to be found that maximize an externally given reward scheme. Section 2 proposes reward-based evaluation framework for

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 127 – 138, 2006.
© Springer-Verlag Berlin Heidelberg 2006

region discovery. In sections 3 and 4 three supervised clustering algorithms are introduced and compared. Section 5 discusses related work and section 6 gives a conclusion. Table 1 summarizes the notations used in this paper.

Table 1. Notations used

Notation	Description
$O=\{o_1, \ldots, o_n\}$	Objects in a dataset (or training set)
n	Number of objects in the dataset
$c_i \subset O$	The i-th cluster
$X=\{c_1, \ldots, c_k\}$	A clustering solution consisting of clusters c_1 to c_k
$q(X)$	Fitness function that evaluates a clustering X
C	A class label

2 Measuring the Interestingness of a Set of Regions

As we explained earlier, our approach uses supervised clustering algorithms to identify interesting regions in a dataset. A region, in our approach, is defined as a surface containing a set of spatial objects; e.g. the convex hull of the objects belonging to a cluster. Moreover, we require regions to be disjoint and contiguous; that is, for each pair of objects belonging to a region, there always must be a path within this region that connects the pair of objects. Furthermore, we assume that the number of regions is not known in advance, and therefore finding the best number of regions is one of the objectives of the clustering process. Therefore, our evaluation scheme has to be capable of comparing clusterings that use a different number of clusters.

Our approach employs a reward-based evaluation framework. The quality q(X) of a clustering X is computed as the sum of the rewards obtained for each cluster $c \in X$. Cluster rewards are weighted by the number of objects that belong to a cluster c. In general, we are interested in finding larger clusters if larger clusters are equally interesting as smaller clusters. Consequently, our evaluation scheme uses a parameter β with $\beta>1$ and fitness increases nonlinearly with cluster-size dependent on the value of β, favoring clusters c with more objects.

$$q(X) = \sum_{c \in X} (reward(c) * |c|^{\beta}) \tag{1}$$

Selecting larger values for the parameter β usually results in a smaller number of clusters in the best clustering X. The proposed evaluation scheme is very general; different reward schemes that correspond to different measures of interestingness can easily be supported in this framework, and the supervised clustering algorithm that will be introduced in the second half of the paper can be run with different fitness functions without any need to change the clustering algorithm itself.

In this paper, due to the lack of space, we only introduce a single measure of interestingness that centers on discovering *hotspots* and *coldspots* in a dataset. The measure is based on a class of interest C, and rewards regions in which the distribution of class C significantly deviates from its prior probability, relying on a reward function τ. τ itself, see Fig. 1, is computed based on p(c,C), prior(C), and based on the following parameters: γ1, γ2, R+, R− with γ1≤1≤γ2; 1≥ R+,R−≥0, η>0 .

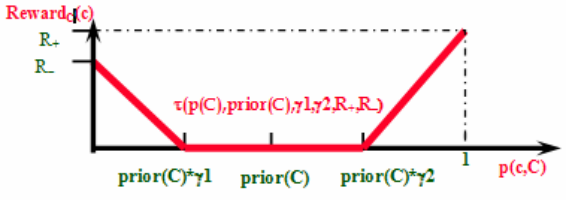


Fig. 1. The reward function τ_C for η=1

The fitness function q(X) is defined as follows:

$$q(X) = \sum_{i=1}^{|X|} \frac{\tau(p(c_i, C), prior(C), \gamma_1, \gamma_2, R+, R-, \eta) * (|c_i|)^{\beta}}{n^{\beta}} \tag{2}$$

with

$$\tau(p(c_i, C), prior(C), \gamma_1, \gamma_2, R+, R-, \eta) = \begin{cases} \left(\frac{((prior(C)*\gamma_1) - P(c_i, C))}{(prior(C)*\gamma_1)} * R-\right)^{\eta} & if \quad P(c_i, C) < prior(C)*\gamma_1 \\ \left(\frac{(P(c_i, C) - (prior(C)*\gamma_2))}{(1-(prior(C)*\gamma_2))} * R+\right)^{\eta} & if \quad P(c_i, C) > prior(C)*\gamma_2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In the above formula prior(C) denotes the probability of objects in dataset belonging to the class of interest C. The parameter η determines how quickly the reward grows to the maximum reward (either R+ or R−). If η is set to 1 it grows linearly; in general, if we are interested in giving higher rewards to purer clusters, it is desirable to choose larger values for η; e.g. η=8.

Let us assume a clustering X has to be evaluated with respect to a class of interest "Poor" with prior(Poor) = 0.2 in a dataset that contains 1000 examples. Suppose that the generated clustering X subdivides the dataset into five clusters c1, c2, c3, c4, and c5 with the following characteristics: |c1| = 50, |c2| = 200, |c3| = 200, |c4| = 350, |c5| = 200; p(c1, Poor) = 20/50, p(c2, Poor) = 40/200, p(c3, Poor) = 10/200, p(c4, Poor) = 30/350, p(c5, Poor) = 100/200. Moreover, the parameters used in the fitness function are as follows: γ1 = 0.5, γ2 = 1.5, R+ = 1, R− = 1, β = 1.1, η=1. Due to the settings of γ1 = 0.5, γ2 = 1.5, clusters that contain between 0.5 x 0.2 = 10% and 1.5 x 0.2 = 30% instances of the class "Poor" do not receive any reward at all; therefore, no reward is given to cluster c2. The remaining clusters received rewards because the distribution

of class "Poor" in the cluster is significantly higher or lower than its prior distribution. For example, the reward for c1 that contains 50 examples is 1/7 x $(50)^{1.1}$; 1/7 is obtained as follows: $\tau(c1) = ((0.4\text{-}0.3)/(1\text{-}0.3)) * 1 = 1/7$.

$$q(X) = \frac{\frac{1}{7} * 50^{1.1} + 0 + \frac{1}{2} * 200^{1.1} + \frac{1}{7} * 350^{1.1} + \frac{2}{7} * 200^{1.1}}{1000^{1.1}} = 0.129$$

3 Supervised Clustering Algorithms for Region Discovery

As part of our research, we have designed and implemented seven supervised clustering algorithms three of which will be described in this Section.

3.1 Supervised Clustering Using Agglomerative Hierarchical Techniques (SCAH)

SCAH is an agglomerative, hierarchical supervised clustering algorithm. Initially, it forms single object clusters, and then greedily merges clusters as long as the clustering quality improves. In more detail, a pair of clusters (c_i, c_j) is considered to be a merge candidate if c_i is the closest cluster to c_j or c_j is the closest cluster to c_j. Distances between clusters are measured by using the average distance between the objects belonging to the two clusters. The pseudo code of the SCAH algorithm is given in Fig. 2.

Inputs:
A dataset $O=\{o_1,...,o_n\}$
A dissimilarity *Matrix* $D = \{d(o_i,o_j) \mid o_i,o_j \in O\}$,
Output:
Clustering $X=\{c_1, c_2, \ldots, c_k\}$

Algorithm:
1) **Initialize**:
   Create single object clusters: $c_i = \{o_i\}, 1 \le i \le n;$
   Compute merge candidates
2) **DO FOREVER**
   a) Find the pair (c_i, c_j) of merge candidates that improves q(X) the most
   b) If no such pair exist terminate, returning $X = \{c_1, c_2, \ldots c_k\}$
   c) Delete the two clusters c_i and c_j from X and add the cluster $c_i \cup c_j$ to X
   d) Update merge candidates

Fig. 2. The SCAH Algorithm

In general, SCAH differs from traditional hierarchical clustering algorithms which merge the two clusters that are closest to each other in that it considers more alternatives for merging clusters. This is important for supervised clustering because merging two regions that are closest to each other will frequently not lead to a better clustering, especially if the two regions to be merged are dominated by instances belonging to different classes.

3.2 Supervised Clustering Using Hierarchical Grid-Based Techniques (SCHG)

Grid-based clustering methods are designed to deal with the large number of data objects in a high dimensional attribute space. A grid structure is used to quantize the space into a finite number of cells on which all clustering operations are performed. The main advantage of this approach is its fast processing time which is typically independent of the number of data objects, and only depends on the number of occupied cells in the quantized space.

SCHG is an agglomerative, grid-based clustering method. Initially, each occupied grid cell is considered to be a cluster. Next, SCHG tries to improve the quality of the clustering by greedily merging two clusters that share a common boundary. The algorithm terminates if q(X) cannot be improved by further merging.

3.3 Supervised Clustering Using Multi-Resolution Grids (SCMRG)

Supervised Clustering using Multi-Resolution Grids (SCMRG) is a hierarchical grid based method that utilizes a divisive, top-down search: each cell at a higher level is partitioned further into a number of smaller cells in the next lower level, but this process only continues if the sum of the rewards of the lower level cells is higher than the obtained reward for the cell at the higher level. The returned cells usually have different sizes, because they were obtained at different level of resolution. The algorithm starts at a user defined level of resolution, and considers three cases when processing a cell:

1. If a cell receives a reward, and its reward is larger than the sum of the rewards associated of its children and larger than the sum of rewards of its grandchildren, this cell is returned as a cluster by the algorithm.
2. If a cell does not receive a reward and its children and grandchildren do not receive a reward, neither the cell nor any of its descendents will be included in the result.
3. Otherwise, all the children cells of the cell are put into a queue for further processing.

The algorithm also uses a user-defined cell size as a depth bound; cells that are smaller than this cell size will not be split any further. The employed framework has some similarity with the framework introduced in the STING Algorithm [13] except that our version centers on finding interesting cells instead of cells that contain answers to a given query, and only computes cell statistics when needed and not in advance as STING does.

4 Experimental Evaluation

4.1 Datasets

In order to study the performance of the clustering algorithms presented in section 3, we conducted experiments on a benchmark consisting of 6 spatial datasets. Table 2

gives a summary for the datasets used. Objects belonging to those data sets consist of longitude and latitude and a non-spatial part which is the class label associated with that object.

Table 2. Datasets used in the benchmark

	Dataset Name	# of objects	# of classes
1	B-Complex9	3,031	2
2	Volcano	1,533	2
3	Earthquake-1	3,161	3
4	Earthquake-10	31,614	3
5	Earthquake-100	316,148	3
6	Wyoming-Poverty	493,781	2

B-Complex9 is a two dimensional synthetic spatial dataset whose examples are distributed having different, well-separated shapes. Earthquake and Volcano are spatial datasets containing the longitude and latitude of earthquakes and volcano eruptions, that are classified based on their severity of the event. Earthquake-1 and Earthquake-10 are smaller datasets contains 1% and 10% of the original data respectively. Wyoming-Poverty is a two dimensional spatial dataset indicating the poverty status of residents of the state of Wyoming based on 2000 census data. For more details about the datasets see [12].

4.2 Experimental Evaluation

The proposed algorithms have been evaluated on a benchmark consisting of the datasets described in section 4.1. We tested the algorithms' capability to identify very pure, potentially very small regions (β=1.01, η=6, γ1=0.5, γ2=1.5, R+=1, R−=1) and to identify larger regions (β=3, η=1, γ1=0.5, γ2=1.5, R+=1, R−=1). Table 3 summarizes the results and Fig. 6 and 7 visualize the result of SCAH, SCHG and SCMRG for the B-Complex9 and Volcano datasets. The result of SCAH on B-Complex9 and Volcano is identical for both set of parameters, so we visualized them only once. Purity and quality of final clustering and the number of clusters obtained are reported for each algorithm-dataset pair. Quality is measured using q(X) that was introduced in Section 2. Purity of a clustering is defined as the number of examples that belong to the most frequent class of a cluster over the total number of examples belonging to clusters. All Experiments were conducted on a DELL D600 workstation with an Intel Pentium M processor 1.80GHz and 1 GB of main memory. SCAH did not succeed in producing results (indicated by DNF in Table 3) within 6 hours for the Earthquake-10 dataset and ran out of storage for the Wyoming and Earthquake-100 datasets when creating the initial clustering.

Table 3. Experimental Results for $\beta = 1.01/\eta = 6$ and $\beta = 3/\eta = 1$

Dataset	Algorithms	SCAH	SCHG	SCMRG	SCAH	SCHG	SCMRG
	Parameters	$\beta = 1.01, \eta = 6$			$\beta = 3, \eta = 1$		
B-Complex9	Purity	1	0.998	1	1	0.997	0.863
	Quality	0.974	0.974	0.957	0.008	0.044	0.002
	Clusters	17	15	132	17	9	22
Volcano	Purity	1	0.692	0.979	1	0.692	0.885
	Quality	0.940	0.091	0.822	1E-5	7E-4	1E-4
	Clusters	639	56	311	639	31	221
Earthquake-1	Purity	1	0.844	0.938	0.853	0.840	0.814
	Quality	0.952	0.399	0.795	0.004	0.086	0.006
	Clusters	479	33	380	161	10	93
Earthquake-10	Purity	DNF	0.840	0.912	DNF	0.834	0.807
	Quality	DNF	0.398	0.658	DNF	0.077	0.006
	Clusters	DNF	37	506	DNF	12	153
Earthquake-100	Purity	DNF	0.842	0.909	DNF	0.837	0.808
	Quality	DNF	0.389	0.560	DNF	0.083	0.006
	Clusters	DNF	38	780	DNF	9	191
Wyoming	Purity	DNF	0.772	0.721	DNF	0.769	0.661
	Quality	DNF	0.027	0.227	DNF	0	0.001
	Clusters	DNF	489	89	DNF	391	78

As can be seen in Table 3, SCAH outperforms SCHG and SCMRG for β=1.01/ η=6, and, in general, performs quite well for small β values, such as 1.01. In general, we observed for these and other datasets that SCAH merges pure clusters that share the same majority class initially. Consequently, it does a quite good job, if the task is to identify small regions that are pure. However, when SCAH reaches the point when it runs out of pure clusters to merge, it has the tendency to terminate prematurely with too small regions. It should be noted that the penalty for having more clusters is quite small if β is 1.01 and the penalty for losing purity when merging clusters is quite high when η is 6. However, for β=3/η=1, the reward gains from having larger clusters are quite significant. This explains why SCAH is outperformed by the other two algorithms for several datasets.

Why does SCAH terminate prematurely when we are interested in obtaining large clusters? The first reason is that SCAH only considers a single merge candidate per cluster whereas SCHG considers four merge candidates per cluster initially; therefore, SCHG has more options for merge, and therefore is less likely to terminate prematurely.

The second reason is that SCAH's look-ahead horizon is too limited. For example, when running SCAH for the B-Complex9 dataset for β=3, the algorithm terminates with 17 clusters, seven of which are depicted in Fig. 3: the outside elliptical shape belongs to class 1 and the two inside spots belong to class 0. Based on average

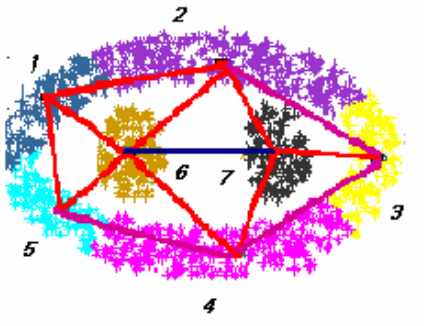
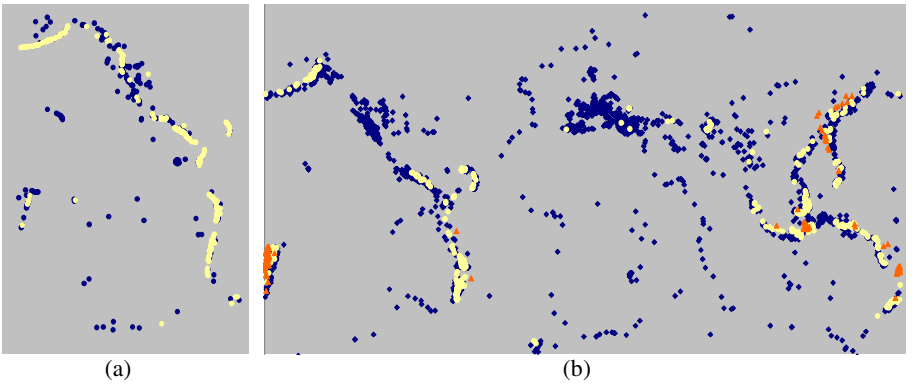


Fig. 3. A part of the B-Complex9 Dataset

distance, the merge candidates considered by SCAH are: 1 and 6, 5 and 6, 3 and 7, 2 and 7, 4 and 7. SCAH will stop here since no improvement made using a single merge. However, for β=3 a higher reward can be obtained by merging all 7 clusters, but SCAH fails to do so, because it terminates if q(X) cannot be improved by a single merge.

Another interesting observation is that the SCHG algorithm outperforms the SCMRG for β=3 for all datasets tested. On the other hand, SCMRG algorithm performs better, compared to SCHG, on the datasets containing chain patterns such as volcano, as depicted in Fig. 4, for β=1.01. This can be attributed to the fact that SCHG is limited to the predefined size of the cells, whereas SCMRG uses grid-cells of different size. As we see in Fig. 6, regions discovered for the volcano dataset are purer than those discovered by the SCHG algorithm.



(a) (b)

Fig. 4. Original Volcano and Earthquake Datasets, a) Part of Volcano b) Earthquake

The average running time of each algorithm is depicted in Fig. 5. The SCAH algorithm is less efficient compared to SCHG and SCMRG. The slowness SCAH can be

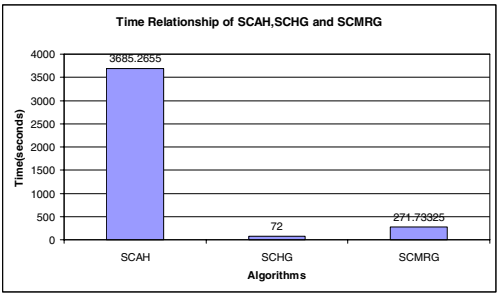


Fig. 5. Average running time of SCAH, SCHG and SCMRG

attributed to its time consuming distance computations: SCAH is required to compute distances between pairs of clusters in order to make the decision of which two clusters have to be merged in the next step, whereas SCHG and SCMRG do not compute any distances at all, and determine merge/split candidates quickly from the underlying grid structure. Moreover, the average running time of SCMRG is higher than SCHG because SCMRG looks for the regions of interest at different levels of resolution, whereas SCHG searches for the interesting clusters using grid cells of fixed size.

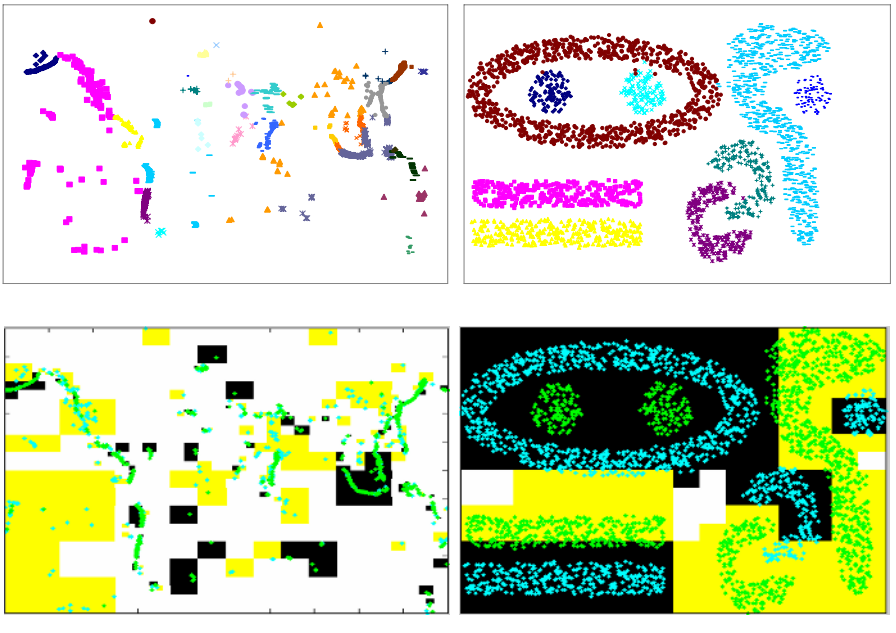


Fig. 6. The result of running supervised algorithms, from top to bottom SCAH, SCHG and SCMRG on two datasets for parameters $\beta = 3/\eta = 1$

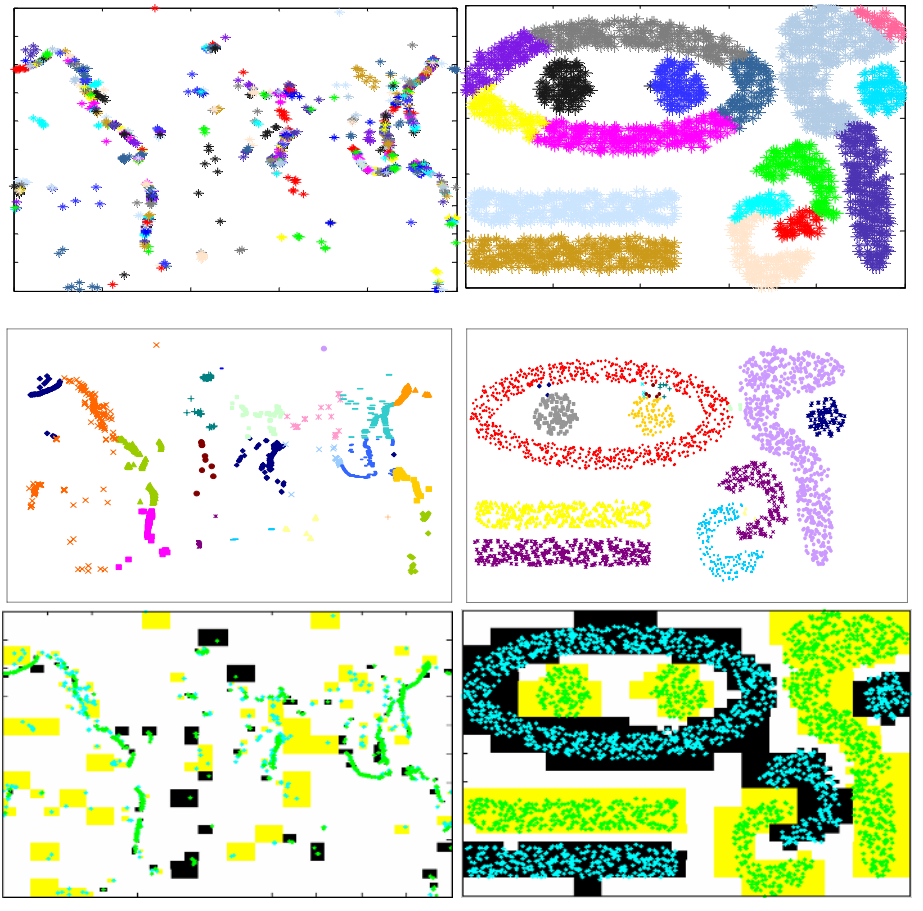


Fig. 7. The result of running SCAH, SCHG and SCMRG on two datasets for $\beta = 1.01/\eta = 6$

5 Related Work

Supervised Clustering [7] centers on partitioning classified examples, maximizing cluster purity while keeping the number of clusters low. Supervised clustering algorithms have been originally proposed to enhance classification algorithms [6]. Our work also has some similarity with work in search-based semi-supervised clustering (also see [1]); for example, Demiriz et al. [5] propose an evolutionary clustering algorithm in which solutions consist of k centroids and the objective of the search process is to obtain clusters that minimize (the sum of) cluster dispersion and cluster impurity. This paper centers on the application of supervised clustering to a new problem: region discovery in spatial datasets containing classified examples.

There also has been some work on co-location rule discovery whose relationship to our work is worth discussing. Co-location rule discovery centers on finding subsets of spatial features frequently located together. Approaches to discover co-location rules

in the literature can be categorized into three groups: spatial statistics [3, 4], association rules [9] and event centric spatial co-location [10]. It should be noted that all mentioned approaches center on finding frequent, global patterns that characterize the complete dataset, whereas our approach centers on finding local regions that are unusual or unexpected with respect to the global characteristics of the dataset.

Hotspot discovery has also been investigated by past research. Williams [14] proposes an evolutionary hotspot discovery architecture that uses traditional clustering, rule induction, and a domain specific fitness function. Tay & Lim et al. [11] describes a region growing method for hotspot discovery, which selects seed points first and then grows clusters from seed points by adding neighbor points as long as a density threshold condition is satisfied. Brimicombe proposes the Geo-ProZone algorithm [2] for hotspot discovery that employs adaptive recursive tessellations. This algorithm supports different level of resolution and recursively decomposes the space with variable decomposition ratios using rectangular grid cells. Finally, Klösgen and M. May [8] propose a multi-relational framework for subgroup discovery within a spatial database system.

6 Summary

In this paper, we introduced a supervised clustering approach and a reward-based evaluation framework for region discovery. Finding interesting regions in spatial datasets is viewed as a clustering problem, in which the sum of rewards for the obtained clusters is maximized, and where the reward associated with a cluster reflects its degree of interestingness for the problem at hand. We explained that this approach is quite different from most other work in spatial data mining that mostly uses association rules. Different measures of interestingness can easily be supported in the proposed framework by designing different reward-based fitness functions; in this case, neither the supervised clustering algorithm itself nor our general evaluation framework has to be modified. We also discussed how hierarchical, and grid-based clustering algorithms can be adapted for supervised clustering in general, and for region discovery in particular, and presented evidence concerning the usefulness of the proposed framework for hotspot discovery problems. Finally, the paper identified some shortcomings of agglomerative clustering algorithms, such as SCAH. Our current work explores the use preprocessing techniques to speed up SCAH, and on the use of proximity graphs to merge clusters more intelligently.

References

1. Basu, S., Bilenko,M., Mooney, R.: Comparing and Unifying Search-based and Similarity-Based Approaches to Semi-Supervised Clustering. in Proc. ICML03 Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, Washington DC (2003) 42-29.
2. Brimicombe, A.J.: Cluster Detection in Point Event Data Having Tendency Towards Spatially Repetitive Events, 8th International Conference on GeoComputation, Ann Arbor, Michigan (2005).

3. Chou, Y.: In Exploring Spatial Analysis in Geographic Information System, Onward Press, (ISBN: 1-56690-119-7), (1997).
4. Cressie, N.: In Statistics for Spatial Data, Wiley-Interscience, (ISBN: 0-471-00255-0), (1993).
5. Demiriz, A., Benett, K.-P., and Embrechts, M.J.: Semi-supervised Clustering using Genetic Algorithms, in Proc. ANNIE, St. Louis, Missouri (1999).
6. Eick, C., Zeidat, N.: Using Supervised Clustering to Enhance Classifiers, in Proc. 15th International Symposium on Methodologies for Intelligent Systems, Saratoga Springs, New York (2005).
7. Eick, C., Zeidat, N., Zhao, Z.: Supervised Clustering - Algorithms and Benefits, UH-Technical Report UH-CS-05-10; short version appeared in Proc. International Conference on Tools with AI (ICTAI), Boca Raton, Florida (2004) 774-776.
8. Klösgen, W. and May, M.: Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database. In Proc. Principles of Data Mining and Knowledge Discovery Conference (PKDD), Springer-Verlag, Berlin (2002) 275-286.
9. Koperski, K., Han, J.: Discovery of Spatial Association Rules in Geographic Spatial Data bases, in Proc. 4th Int. Symp. Advances in Spatial Databases, Maine (1995) 47-66.
10. Shekhar, S., Huang, Y.: Discovering Spatial Co-location Patterns: A Summary of Results, in Proc. of 7th International Symposium on Spatial and Temporal Databases (SSTD01), L.A., CA (2001).
11. Tay S.C., Lim, K.H., Yap, L.C.: Spatial Data Mining: Clustering of Hot Spots and Pattern Recognition, IEEE (2003).
12. UH Data Mining & Machine Learning Group: http://www.tlc2.uh.edu/dmmlg/Datasets.
13. Wang, W., Yang, J., Muntz, R.: STING: A Statistical Information Grid Approach to Spatial Data Mining, Proceedings of the 23rd VLDB Conference, Athens, Greece (1997).
14. Williams, G.J.: Evolutionary Hot Spots Data Mining, An Architecture for Exploring for Interesting Discoveries, Pacific Asia Conference on Knowledge Discovery and Data Mining, Beijing, China (1999).

Optimal String Mining Under Frequency Constraints

Johannes Fischer[1], Volker Heun[1], and Stefan Kramer[2]

[1] Ludwig-Maximilians-Universität München, Institut für Informatik,
Amalienstr. 17, D-80333 München
{Johannes.Fischer, Volker.Heun}@bio.ifi.lmu.de
[2] Technische Universität München, Institut für Informatik/I12,
Boltzmannstr. 3, D-85748 Garching b. München
kramer@in.tum.de

Abstract. We propose a new algorithmic framework that solves frequency-related data mining queries on databases of strings in optimal time, i.e., in time linear in the input and the output size. The additional space is linear in the input size. Our framework can be used to mine frequent strings, emerging strings and strings that pass other statistical tests, e.g., the χ^2-test. In contrast to the presented result for strings, no optimal algorithms are known for other pattern domains such as itemsets. The key to our approach are several recent results on index structures for strings, among them suffix- and lcp-arrays, and a new preprocessing scheme for range minimum queries. The advantages of array-based data structures (compared with dynamic data structures such as trees) are good locality behavior and extensibility to secondary memory. We test our algorithm on real-world data from computational biology and demonstrate that the approach also works well in practice.

1 Introduction

In many applications, e.g., in computational biology, the goal is to find interesting string or sequence patterns in data. Application areas are, among others, finding discriminative features for sequence classification or segmentation [1], discovering new binding motifs of transcription factors, or probe design [2]. In this paper, we focus on string mining under frequency constraints, i.e., predicates over patterns depending solely on the frequency of their occurrence in the data. This category encompasses combined minimum/maximum support constraints, constraints concerning emerging substrings, and constraints concerning statistically significant substrings. We present an algorithm that is able to answer such queries optimally, that is, in time linear in the size of the input database, plus the time to output the solution patterns.

In previous work [2], we investigated string mining approaches based on breakthrough results on index structures for strings, among them suffix arrays and longest common prefix (lcp) tables [3,4,5]. Suffix arrays are essentially a representation of the lexicographic order of all suffixes of a string. lcp tables contain the

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 139–150, 2006.
© Springer-Verlag Berlin Heidelberg 2006

length of the longest common prefix of two consecutive suffixes in a suffix array. For both suffix arrays and lcp tables, fast construction algorithms are known. As in our previous approach, we assume that the suffix array and the lcp table are computed in a preprocessing step. The key to the approach presented here is another preprocessing scheme for so-called range minimum queries (RMQs). RMQs generalize the lcp table in the sense that the length of the longest common prefix can be answered for *arbitrary* suffixes. Taking advantage of recent results [6,7], it is possible to answer RMQs in constant time. Another technical novelty is the solution to computing the frequency counts. The solution first determines the number of all occurrences (counting several occurrences per example), and then subtracts so-called correction terms to obtain the final counts per example. It is shown that the presented approach is able to answer all frequency-related constraints in linear time, i.e., optimally. For instance, it is possible to compute all statistically significant substrings between two classes of strings in time linear in the total length of the strings in the database, plus the total length of all such significant strings (i.e., the output size). It is interesting to note that no optimality results are known for other pattern domains such as itemsets or graphs (see, e.g., [8]).

While the focus of this paper lies on the algorithm and the theoretical result, we also implemented and tested the approach to show that it works in practice. In our experiments, we compared protein sequences from humans and mice, in total more than 40MB of sequence data. The aim of the experiments was to mine all *frequent substrings* (Probl. 1, Sect. 2) and *emerging substrings*, respectively (Probl. 2). The only known algorithm for emerging substrings [9] runs in quadratic time, and is therefore not applicable. The experiments confirm that our approach works well in practice. In particular, most queries for emerging substrings can be answered in less than three minutes, and the mining of frequent substrings is 2–3 times faster than our previous method presented in [2].

2 Preliminaries

We consider patterns from the domain of strings. For a finite ordered alphabet Σ, a string ϕ is a chain $\phi_1 \dots \phi_n$ of letters $\phi_i \in \Sigma$. We often write $\phi_{n..m}$ to denote the substring of ϕ ranging from position n to m. $|\phi|$ denotes the number of letters in ϕ. $\Sigma^\star$ is the set of all strings over Σ. For $\phi, \psi \in \Sigma^\star$ we write $\phi \preceq \psi$ if ϕ is a substring of ψ. $\mathrm{lcp}(\phi, \psi)$ gives the *length* of the *longest common prefix* of ϕ and ψ. For example, $\mathrm{lcp}(\mathtt{aab}, \mathtt{abab}) = 1$. Given a database $\mathcal{D} \subseteq \Sigma^\star$ with strings over Σ, we write $|\mathcal{D}|$ to denote the number of strings in $\mathcal{D}$, and $\|\mathcal{D}\|$ to denote their total length, i.e., $\|\mathcal{D}\| = \sum_{\phi \in \mathcal{D}} |\phi|$. We define the *frequency* and the *support* of a pattern $\phi \in \Sigma^\star$ in $\mathcal{D}$ as follows:

$$\mathsf{freq}(\phi, \mathcal{D}) := |\{d \in \mathcal{D} \ : \ \phi \preceq d\}|, \ \mathsf{supp}(\phi, \mathcal{D}) := \frac{\mathsf{freq}(\phi, \mathcal{D})}{|\mathcal{D}|}$$

Note that this is not the same as counting all occurrences of a ϕ in $\mathcal{D}$, because one database entry could contain multiple occurrences of ϕ. The main contribution

of this article is to show how one can compute the frequency (or support) of all strings occurring at least once in one of the databases in optimal time, i.e., in time linear in the size of the input databases. This allows us to solve frequency-related mining queries in optimal time, i.e., in time linear in the sum of the input- and the output-size. Naturally, the query must be computable from the frequency (or support) in constant time.

We now introduce three problems that can be solved optimally with our approach. The first one is as follows.

Problem 1. Given m databases $\mathcal{D}_1, \ldots, \mathcal{D}_m$ of strings over Σ and m pairs of frequency thresholds $(min_1, max_1), \ldots, (min_m, max_m)$, the *Frequent Pattern Mining Problem* is to return all strings $\phi \in \Sigma^\star$ that satisfy $min_i \leq \mathsf{freq}(\phi, \mathcal{D}_i) \leq max_i$ for all $1 \leq i \leq m$.

This well-known problem has been addressed by many authors using different solution strategies and data-structures ([10,11,2]), but none of these is optimal.

Next, we consider a 2-class problem for a (usually positive) database $\mathcal{D}_1$ and a (usually negative) database $\mathcal{D}_2$. We define the *growth-rate* from $\mathcal{D}_2$ to $\mathcal{D}_1$ of a string ϕ as

$$\mathsf{growth}_{\mathcal{D}_2 \to \mathcal{D}_1}(\phi) := \frac{\mathsf{supp}(\phi, \mathcal{D}_1)}{\mathsf{supp}(\phi, \mathcal{D}_2)}, \text{ if } \mathsf{supp}(\phi, \mathcal{D}_2) \neq 0 \ ,$$

and $\mathsf{growth}_{\mathcal{D}_2 \to \mathcal{D}_1}(\phi) = \infty$ otherwise. The following definition is motivated by the problem of mining Emerging Patterns [12]:

Problem 2. Given two databases $\mathcal{D}_1$ and $\mathcal{D}_2$ of strings over Σ, a support threshold ρ_s $(0 < \rho_s \leq 1)$, and a minimum growth rate $\rho_g > 1$, the *Emerging Substrings Mining Problem* is to find all strings $\phi \in \Sigma^\star$ such that $\mathsf{supp}(\phi, \mathcal{D}_1) \geq \rho_s$ and $\mathsf{growth}_{\mathcal{D}_2 \to \mathcal{D}_1}(\phi) \geq \rho_g$.

The patterns satisfying both the support- and the growth-rate condition are called *Emerging Substrings* (ESs). ESs with an infinite growth-rate are called *Jumping Emerging Substrings* (JESs), because they are highly discriminative for the two databases. The only known solution for finding ESs [9] is quadratic in the input size. The following example will be continued throughout this paper.

Example 1. Let $\mathcal{D}_1 = \{\mathtt{aaba}, \mathtt{abaaab}\}$, $\mathcal{D}_2 = \{\mathtt{bbabb}, \mathtt{abba}\}$, $\rho_s = 1$, and $\rho_g = 2$. Then the emerging substrings from $\mathcal{D}_2$ to $\mathcal{D}_1$ are `aa`, `aab` and `aba`. In this case, these are also the jumping ESs. ◇

As a last example problem that our method can solve optimally we mention the χ^2-test.

Problem 3. Given m databases $\mathcal{D}_1, \ldots, \mathcal{D}_m$ of strings over Σ and a threshold ρ. Let $n = \sum_{j=1}^{m} |\mathcal{D}_j|$ be the total number of strings, $f = \sum_{i=1}^{m} \mathsf{freq}(\phi, \mathcal{D}_i)$ the total frequency of ϕ, and $\mathbf{E}_j = f \cdot |\mathcal{D}_j|/n$ be the expected value of ϕ's frequency. Then ϕ is significant if it passes the χ^2-test, i.e., if $\chi^2 = \sum_{j=1}^{m} \frac{(\mathsf{freq}(\phi, \mathcal{D}_j) - \mathbf{E}_j)^2}{\mathbf{E}_j} \geq \rho$.

2.1 Suffix- and lcp-Arrays

This section introduces two fundamental data structures that we need for our algorithm. We write $A[1,n]$ for an array A of length n, and $A[i]$ denotes the i'th entry of A. To make the following definitions as general as possible, let t denote an arbitrary string of length n. Later, t will be formed from the input databases (fully explained in Sect. 3.1). Recall that $t_{i..j}$ is the substring from i to j.

The *suffix array* SA (see [3,4]) for t is used to describe the lexicographic order of t's suffixes, in the sense that it "enumerates" the suffixes from the smallest to the largest. More formally, $\mathsf{SA}[1,n]$ is an array of integers s.t. its entries contain all of the numbers from 1 to n (i.e., $\{\mathsf{SA}[1],\ldots,\mathsf{SA}[n]\} = \{1,\ldots,n\}$), and $t_{\mathsf{SA}[i]..n}$ is lexicographically less than $t_{\mathsf{SA}[i+1]..n}$ for all $1 \leq i < n$. See Fig. 1(a) for an example, which builds on the databases $\mathcal{D}_1$ and $\mathcal{D}_2$ from Ex. 1.

The suffix array for t can be computed in $O(n)$ time, either indirectly by constructing a suffix tree for t, or directly with some recent methods, e.g. [13]. In practice, however, asymptotically slower algorithms [14,15] have been shown to perform faster. The method in [15] has the further advantage that it uses only ϵn additional bytes of space, which is close to optimal. Here, ϵ is a tunable parameter that determines the speed of the algorithm and can be made arbitrarily small.

The lcp-array $\mathsf{LCP}[1,n]$ for t is defined by $\mathsf{LCP}[i] = \mathrm{lcp}(t_{\mathsf{SA}[i]..n}, t_{\mathsf{SA}[i-1]..n})$ for all $1 < i \leq n$, and $\mathsf{LCP}[1] = 0$. That is, LCP contains the lengths of the longest common prefixes of t's suffixes that are consecutive in lexicographic order. Kasai et al. [5] gave an algorithm to compute LCP in $O(n)$ time, and Manzini [16] adapted this algorithm to use only one integer array. It can be argued that most of the LCP-values are small compared with the size of the text and could thus be stored in less than n words, but we do not pursue this approach here.

2.2 Range Minimum Queries

The last tool we need for our linear-time approach is a preprocessing of the lcp-array such that *range minimum queries* (RMQs) can be answered in constant time. The reason for using RMQs on LCP is that they generalize the lcp-array, in the sense that we can compute the lcp between arbitrary suffixes, and not only between those that are lexicographically adjacent. Formally, for two given indices i and j the query $\mathrm{RMQ}_{\mathsf{LCP}}(i,j)$ asks for the position of the minimum element in $\mathsf{LCP}[i,j]$, i.e., $\mathrm{RMQ}_{\mathsf{LCP}}(i,j) := \arg\min_{k\in\{i,\ldots,j\}}\{\mathsf{LCP}[k]\}$. We return the smallest index if the minimum is not unique.

Lemma 1. *Let $t \in \Sigma^\star$ be a text and LCP be the lcp-array for t. Then for all $1 \leq i < j \leq |t|$, $\mathrm{lcp}(t_{\mathsf{SA}[i]..|t|}, t_{\mathsf{SA}[j]..|t|})$ is given by $\mathsf{LCP}[\mathrm{RMQ}_{\mathsf{LCP}}(i+1,j)]$.*

This follows immediately from the definition of the lcp-array. Stated differently, Lemma 1 says that the i'th- and the j'th-smallest suffix of t are equal in exactly their $\mathsf{LCP}[\mathrm{RMQ}_{\mathsf{LCP}}(i+1,j)]$ first characters.

It has been shown that a linear preprocessing of any input array A is sufficient to find $\mathrm{RMQ}_A(i,j)$ in time $O(1)$ [6]. This method has recently been refined

to use only $o(n)$ extra space [7]. The basic idea for both approaches is to divide the array into blocks of size $\Theta(\log n)$. Then each block is preprocessed such that a query that lies completely *inside* one block can be answered in constant time. This step is accomplished by applying the so-called Four-Russians-Trick [17] to the blocks (precomputation of all results for sufficiently small instances). A final step preprocesses the array such that queries that exactly span over several blocks can be answered efficiently. In total, each range minimum query can be decomposed into at most three sub-queries, where the first and the last of these are in-block-queries, and the second is an out-of-block-query. As each sub-query can be answered in constant time, the overall query time is $O(1)$ [6,7].

3 The New Algorithm

In this section we present our linear-time algorithm for answering frequency-related mining queries (e.g., emerging substrings). Logically, the algorithm can be divided into three main phases: (1) Preprocessing, (2) Labeling, and (3) Extraction. The preprocessing step constructs all necessary data structures: the suffix- and lcp-array, and the preprocessing for RMQ. The labeling step does the principal work for a fast calculation of the string-frequencies. Finally, the extraction step returns all strings passing the frequency-based criterion.

The main idea for computing the frequencies is as follows. Let $\mathcal{D}_j = \{s^{j,1}, \ldots, s^{j,|\mathcal{D}_j|}\}$ be the given databases ($1 \leq j \leq m$). For the strings ϕ occurring in any of the databases, we compute the *total* number of occurrences in $\mathcal{D}_j$ and store the respective numbers in $S_{\mathcal{D}_j}(\phi)$. We further compute so-called *correction terms* $C_{\mathcal{D}_j}(\phi)$ that count how often string ϕ has a repetition in the *same* string of $\mathcal{D}_j$:

$$S_{\mathcal{D}_j}(\phi) = |\{(i,k) : s^{j,k}_{i..i+|\phi|-1} = \phi\}|,\ C_{\mathcal{D}_j}(\phi) = \sum_{k=1}^{|\mathcal{D}_j|} (|\{i : s^{j,k}_{i..i+|\phi|-1} = \phi\}| - 1) \quad (1)$$

Then $\mathsf{freq}(\phi, \mathcal{D}_j)$ clearly equals $S_{\mathcal{D}_j}(\phi) - C_{\mathcal{D}_j}(\phi)$. In our example, $S_{\mathcal{D}_1}(\mathtt{ab}) = 3$ (there are 3 occurrences of `ab` in $\mathcal{D}_1$) and $C_{\mathcal{D}_1}(\mathtt{ab}) = 1$ (`ab` is repeated once in the second string $s^{1,2} = \mathtt{abaaab}$ of $\mathcal{D}_1$). We will see in Sect. 3.3 that it is not very hard to compute the S-numbers in linear time; the real difficulty lies in the computation of the C-numbers. The following lemma suggests how the lcp-array can be used to calculate these correction terms (cf. Lemma 1):

Lemma 2. *For any string ϕ occurring in $\mathcal{D}_j$, $C_{\mathcal{D}_j}(\phi)$ is given by the number of times that ϕ is a prefix of the longest common prefix of two lexicographically adjacent suffixes from the same string $s^{j,k}$ in $\mathcal{D}_j$:*

$$C_{\mathcal{D}_j}(\phi) = \sum_{k=1}^{|\mathcal{D}_j|} |\{(i,i') : \phi \text{ prefix of lex. adj. suffixes of } s^{j,k} \text{ starting at } i \neq i'\}|$$

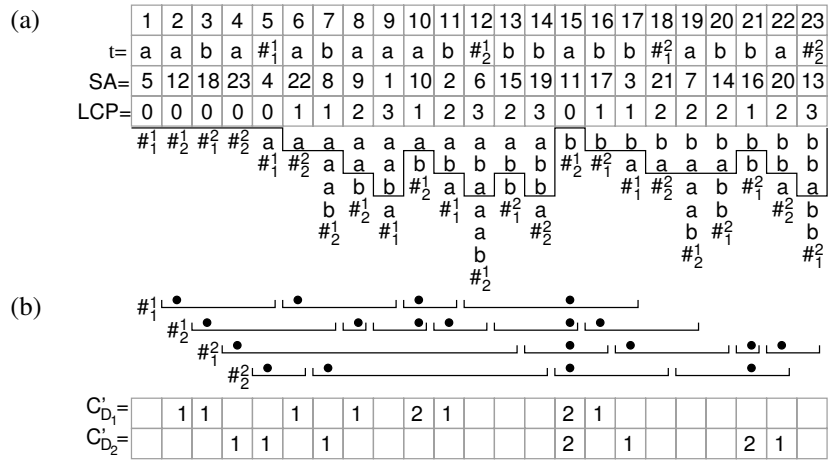


Fig. 1. (a) The suffix array for w and its lcp-table. Below position i we draw the string $t_{\mathsf{SA}[i]..n}$ until reaching the first end-of-string marker. The solid line going through these strings indicates the lcp-values. (b) Computation of the C -numbers. The intervals are those for which range minimum queries on LCP are executed; the position of the minimum is depicted by a solid circle. Empty fields in the two arrays denote 0.

We omit the proof of this result due to space limitations. As an example, the suffixes of $s^{1,2} = \texttt{abaaab}$ are (in lexicographic order) `aaab`, `aab`, `ab`, `abaaab`, `b`, and `baaab`. The third and the fourth have `ab` as their longest common prefix. Because no suffixes from $s^{1,1} = \texttt{aaba}$ have `ab` as their longest common prefix, the value of $C_{\mathcal{D}_1}(\texttt{ab})$ is 1.

The calculation of the correction terms is done in phase (2) and (3) of our algorithm. In phase (2), we create auxiliary arrays that allow an easy computation of the actual correction terms. The computation of the C-terms is then done along with the computation of the S-numbers in phase (3). The following sections describe the three phases in greater detail.

3.1 Preprocessing

We form a (conceptual) string $s^{1,1}\#_1^1 \ldots s^{1,|\mathcal{D}_1|}\#_{|\mathcal{D}_1|}^1 \ldots s^{m,1}\#_1^m \ldots s^{m,|\mathcal{D}_m|}\#_{|\mathcal{D}_m|}^m$ which we denote by t. The $\#_i^k$'s are new symbols that do not occur in any of the databases and serve to mark the end of a string from the respective database. Note that the length of t is $n := \sum_{j=1}^{m}(\|\mathcal{D}_j\| + |\mathcal{D}_j|)$. The preprocessing then consists of constructing the following data structures for t (in this order): the suffix array SA, the lcp-array LCP, and the information to answer $\text{RMQ}_{\mathsf{LCP}}(i,j)$ in $O(1)$. All steps take time $O(n)$. See Fig. 1(a) for an example.

A short definition is necessary at this point: We say that entry $\mathsf{SA}[i]$ *points* to string $s^{j,k}$ in database $\mathcal{D}_j$ iff the first end-of-string marker in $t_{\mathsf{SA}[i]..n}$ is $\#_k^j$. For example, in Fig. 1(a), $\mathsf{SA}[8] = 9$ points to $s^{1,2}$, because $t_{9..23} = \texttt{aab}\#_2^1\texttt{bb}\ldots$.

Algorithm 1. Labeling of the lcp-array.

Input. suffix array SA and lcp-array LCP of size n for m databases $\mathcal{D}_1, \ldots, \mathcal{D}_m$
Output. m arrays $C_1, \ldots, C_m$ of size n

```
1  Let last_i be an array of size |D_i|, initialized with all 0 (i = 1,...,m);
2  Let C_i be an array of size n, initialized with all 0 (i = 1,...,m);
3  for i = 1,...,n do
4      Let j and k be defined such that SA[i] points to s^{j,k};
5      if last_j[k] ≠ 0 then
6          l ← RMQ_LCP(last_j[k] + 1, i);
7          increase C_j[l] by 1;
8      end
9      last_j[k] ← i;
10 end
```

3.2 Labeling

Alg. 1 augments the lcp-array LCP with arrays $C'_{\mathcal{D}_1}, \ldots, C'_{\mathcal{D}_m}$ which facilitate the computation of the correction terms in phase 3. Although the $C'_{\mathcal{D}_j}$'s are represented by new arrays of size n, we call this step "labeling" because it is derived from the *tree labeling technique* by Hui [18]. We want $C'_{\mathcal{D}_j}[i]$ to be equal to the number of lexicographically adjacent suffixes from the *same* string in $\mathcal{D}_j$ that share a longest common prefix of length $\mathsf{LCP}[i]$. More formally, $C'_{\mathcal{D}_j}[i]$ equals the number of triples (a, b, k) that fulfill the following constraints:

1. $1 \leq a < i \leq b \leq n$, and $\mathsf{SA}[a]$ and $\mathsf{SA}[b]$ point to the same string $s^{j,k}$ in $\mathcal{D}_j$.
2. No entry strictly between a and b points to $s^{j,k}$.
3. $\mathrm{lcp}(t_{\mathsf{SA}[a]..n}, t_{\mathsf{SA}[b]..n|}) = \mathsf{LCP}[i]$.

Note that point 3 actually states that two lexicographically consecutive suffixes of $s^{j,k}$ have an lcp-value of $\mathsf{LCP}[i]$, because of 1 and 2. Note also that due to the definition of the lcp-array (lengths of longest common prefix of lexicographically adjacent suffixes) there *must* be an $i \in [a, b]$ with $\mathsf{LCP}[i] = \mathrm{lcp}(t_{\mathsf{SA}[a]..n}, t_{\mathsf{SA}[n]..n|})$, so $C'_{\mathcal{D}_j}[i'] \neq 0$ for at least one i' between a and b.

The C'-numbers are computed as follows: the for-loop (lines 3–10) scans the lcp-array from left to right. Array $last_{\mathcal{D}_j}[k]$ holds the rightmost position in SA to the left of i that points to $s^{j,k}$. Thus, if $\mathsf{SA}[i]$ points to $s^{j,k}$, setting $a = last_{\mathcal{D}_j}[k]$ and $b = i$ fulfills constraints 1 and 2 above. Because of Lemma 1, $\mathrm{lcp}(t_{\mathsf{SA}[a]..n}, t_{\mathsf{SA}[b]..n|})$ is given by $\mathrm{RMQ}_{\mathsf{LCP}}(last_{\mathcal{D}_j}[k] + 1, i)$ (line 6). See Fig. 1(b) for an example.

We now sketch how the C'-numbers help to compute the actual correction terms. We compute $C(\phi)$ for the strings ϕ that are *maximally repeated* (also called *branching* in [5]), which means that they occur more than once in t, say x times, but all extensions of ϕ (i.e., strings of which ϕ is a proper prefix) occur less than x times.[1] The number of such strings is clearly linear, and the frequency of all other strings can be derived from one of the maximally repeated strings.

[1] Note that the maximally repeated strings are exactly those strings that correspond to an internal node in the suffix tree [19] for t.

Lemma 3. *Let $\phi \preceq t$. The following is equivalent:*

1. *ϕ is maximally repeated.*
2. *There exist $1 \leq l \leq r \leq n$ s.t.*
   (a) $\mathsf{LCP}[l-1] < \mathsf{LCP}[l]$ and $\mathsf{LCP}[r] > \mathsf{LCP}[r+1]$,
   (b) $\mathsf{LCP}[i] \geq |\phi|$ for all $l \leq i \leq r$,
   (c) $\exists\ q \in \{l, \ldots, r\}$ with $\mathsf{LCP}[q] = |\phi|$ and $\phi = t_{\mathsf{SA}[q]..\mathsf{SA}[q]+|\phi|-1}$.

Parts (a) and (b) say that (l, r) is a maximal interval in SA where all suffixes have a common prefix (namely ϕ), and (c) says that at least two of the suffixes in this interval differ after position $|\phi|$. We refer the interested reader to [20] for a proof of this non-trivial result. From now on, we call (l, r) an *lcp-interval* representing string ϕ if it fulfills the conditions of Lemma 3. A *child-interval* of (l, r) is a maximal proper sub-interval of (l, r) that represents a different string. E.g., in Fig. 1(a), the lcp-interval representing `a` is $(6, 14)$, which has the child-intervals $(8, 9)$ (representing `aa`) and $(11, 14)$ (representing `ab`).

Now, if (l, r) is the lcp-interval that represents ϕ, with Lemma 2 we see that

$$C_{\mathcal{D}_j}(\phi) = \sum_{l \leq i \leq r} C'_{\mathcal{D}_j}[i] = \sum_{\substack{l \leq i \leq r \\ \mathsf{LCP}[i] = |\phi|}} C'_{\mathcal{D}_j}[i] \ + \sum_{\substack{(l',r') \text{ child-interval of } (l,r) \\ (l',r') \text{ represents } \psi \neq \phi}} C_{\mathcal{D}_j}(\psi) \ . \quad (2)$$

(The last step is to enable a recursive calculation of the C-terms.)

Example 2. In Fig. 1, the interval $(8, 9)$ (representing `aa`) gives $C_{\mathcal{D}_1}(\mathtt{aa}) = \sum_{8 \leq i \leq 9} C'_{\mathcal{D}_1}[i] = 1 + 0 = 1$, and the interval $(11, 14)$ (representing `ab`) gives $C_{\mathcal{D}_1}(\mathtt{ab}) = 1 + 0 + 0 + 0 + 0 = 1$. Having this, we can compute $C_{\mathcal{D}_1}(\mathtt{a})$ as $C'_{\mathcal{D}_1}[6] + C'_{\mathcal{D}_1}[7] + C_{\mathcal{D}_1}(\mathtt{aa}) + C'_{\mathcal{D}_1}[10] + C_{\mathcal{D}_1}(\mathtt{ab}) = 1 + 0 + 1 + 2 + 1 = 5$. ⋄

Kasai et al. [5] gave an algorithm that simulates a bottom-up-traversal of the suffix tree by scanning the lcp-array from left to right. We could thus calculate the C-numbers by a modification of their algorithm, applying (2) to all lcp-intervals in a bottom-up manner. However, this step can be incorporated into the extraction step (which we explain next), thereby avoiding the need to store the C-numbers in separate arrays.

3.3 Extraction

We now describe how to output all strings that pass the frequency-based criterion. As mentioned above, this step is accomplished by a simulated depth-first-traversal of the suffix tree [5], calculating for each lcp-interval representing string ϕ the values $S_{\mathcal{D}_j}(\phi)$ and $C_{\mathcal{D}_j}(\phi)$ for $j = 1, \ldots, m$, thereby yielding the frequency of ϕ in $\mathcal{D}_j$ as $S_{\mathcal{D}_j}(\phi) - C_{\mathcal{D}_j}(\phi)$. The formula for the C-numbers is given by (2), and for the S-numbers we have $S_{\mathcal{D}_j}(\phi) = \sum_{\substack{l-1 \leq i \leq r \\ \mathsf{SA}[i] \text{ points to } \mathcal{D}_j}} 1$ (again, (l, r) is ϕ's lcp-interval). As in (2), this can be rewritten to allow a recursive calculation.

Alg. 2 is used for the extraction phase. If one deletes lines 5, 17 and 19 from Alg. 2 and substitutes lines 8–13 by the single command "print $t_{\mathsf{SA}[i]..\mathsf{SA}[i]+v.h-1}$",

Algorithm 2. Extraction of all substrings satisfying p.

Input. suffix array SA, lcp-array LCP, C_j as computed by Alg. 1 (all of size n), frequency-based predicate $p(\mathsf{supp}_1, \ldots, \mathsf{supp}_m)$
Output. All substrings satisfying p

1 S is a stack holding tuples v of the form $(v.h, v.S_j, \ldots, v.S_m, v.S_2, v.C_1, \ldots, v.C_m)$
2 Let v be a stopper element with $v.h = -\infty$, push v on S
3 **for** $i = 1, \ldots, n+1$ **do**
4 $\quad v \leftarrow \mathtt{top}(S)$ {v represents the string to be examined next}
5 $\quad S_j \leftarrow 0$ for all $j = 1, \ldots, m$
6 $\quad$ **while** $v.h > \mathsf{LCP}[i]$ **do**
7 $\quad\quad v \leftarrow \mathtt{pop}(S), w \leftarrow \mathtt{top}(S)$ {w points to top of stack throughout the loop}
8 $\quad\quad$ **if** $w.h \geq \mathsf{LCP}[i]$ **then** $w.S_j$ += $v.S_j$, $w.C_j$ += $v.C_j$ for all j
9 $\quad\quad \mathsf{supp}_j \leftarrow \frac{v.S_{\mathcal{D}_j} \quad v.C_{\mathcal{D}_j}}{j}$ (for all $j = 1, \ldots, m$)
10 $\quad\quad$ **if** $p(\mathsf{supp}_1, \ldots, \mathsf{supp}_m)$ **then**
11 $\quad\quad\quad$ **for** $h = \max\{w.h, \mathsf{LCP}[i]\} + 1, \ldots, v.h$ **do** print $t_{\mathsf{SA}[i]..\mathsf{SA}[i]+h-1}$
12 $\quad\quad$ **end**
13 $\quad\quad S_j \leftarrow v.S_j$ for all $j = 1, \ldots, m$
14 $\quad\quad v \leftarrow w$
15 $\quad$ **end**
16 $\quad$ **if** $v.h < \mathsf{LCP}[i]$ **then** push $(\mathsf{LCP}[i], S_1, \ldots, S_m, 0, \ldots, 0)$ on S
17 $\quad \mathtt{top}(S).C_j$ += $C_j[i]$ for all $j = 1, \ldots, m$
18 $\quad$ **if** $i \leq n$ **then**
19 $\quad\quad$ Let $\mathsf{SA}[i]$ point to $\mathcal{D}_j$; set $S_j \leftarrow 1$ and all other $S_{j'}$'s to 0
20 $\quad\quad$ push $(n - \mathsf{SA}[i] + 1, S_1, \ldots, S_m, 0, \ldots, 0)$ on S
21 $\quad$ **end**
22 **end**

this yields exactly the algorithm in Fig. 7 of [5] which solves the *substring traversal problem*, i.e., the enumeratation of all maximally repeated substrings. The idea behind this algorithm is to visit all suffixes of t in lexicographic order and to keep all maximally repeated prefixes of the current suffix on a stack S, ordered by their length with the longest being on top. A more formal desciption is as follows. Each element on S is represented by a tuple $(h, S_{\mathcal{D}_1}, \ldots, S_{\mathcal{D}_m}, C_{\mathcal{D}_1}, \ldots, C_{\mathcal{D}_m})$, where h is the length of the prefix (i.e., the corresponding prefix is $t_{\mathsf{SA}[i]..v.h-1}$), and the other variables are the counters as defined by (1) (page 143). At the beginning of step i of the for-loop (lines 3–22), we have that the $(i-1)$'th suffix and all maximally repeated prefixes of $t_{\mathsf{SA}[i-1]..n}$ are on S. Then the $(i-1)$'th suffix is visited (line 4) and the following steps are performed:

1. The while-loop (lines 6–15) removes from S all tuples representing strings with length at least $\mathrm{lcp}(t_{\mathsf{SA}[i-1]..n}, t_{\mathsf{SA}[i]..n}) = \mathsf{LCP}[i]$. These are exactly the prefixes of $t_{\mathsf{SA}[i-1]..n}$ which are not a prefix of $t_{\mathsf{SA}[i]..n}$. All strings passing the statistical criterion are returned (line 11).

2. The counter-values $S_{\mathcal{D}_j}(\phi)$ and $C_{\mathcal{D}_j}(\phi)$ of the current string v are added to the respective counters of the string on top of the stack (line 8). This step takes care of the last sum in (2), as v represents a child of the string on top.
3. When pushing the longest common prefix of two lexicographically adjacent suffixes on S (line 16), the counter-values are initialized correctly.
4. The C'-numbers are added to the correct string (line 17) which is again on top of the stack. This step takes care of the first sum in (2).
5. The suffix $t_{\mathsf{SA}[i]..n}$ is pushed on S with the correct counter-values (lines 18–21). Line 19 accounts for the initialisation of the $S_{\mathcal{D}_j}$-values.

It is shown in [5] that this algorithm visits all maximally repeated substrings of t, and its running time is $O(n)$ (apart from the for-loop that outputs the solutions, line 11). The discussion from Sect. 3.2 shows that the S- and C-values are calculated correctly, and thus in line 9 we have that the support of the string ϕ that is represented by v is calculated correctly. We thus have the following

Theorem 1. *For m databases of strings of total length n, all strings that satisfy a frequency-based criterion (e.g., emerging substrings) can be calculated in time $O(n + s)$, where s is the total size of the strings that satisfy the criterion.*

4 Practical Performance

The aim of this section is to show that our new method also works fast in practice, even on large datasets. We implemented the algorithm from Sect. 3 in C++ (available at `www.bio.ifi.lmu.de/~fischer/`) so that it finds emerging substrings (Problem 2) and frequent substrings (Problem 1), respectively. For the construction of SA we used the method presented in [15]. We used two datasets consisting of the primary structure of all protein data from human and mouse, which were obtained from Swissprot using the keywords HUMAN and MOUSE in the NEWT taxonomy browser [21]. The human dataset contained 57,020 proteins of total length ≈23MB, and the mouse dataset contained 50,680 proteins of total length ≈22MB. Because the implementation of the emerging-substring-miner from [9] is not publicly available we could not compare against their method. However, due the sheer size of the input (more than 45MB) it is very unlikely that their quadratic-time approach would work well. As an example, their fastest method takes 20–40 seconds to mine data of approximately 2MB total size, depending on the input parameters [9].

We ran several tests on an Athlon XP 3000 with 2GB of RAM under Linux. We redirected all output to the null-device in order to remove the influence of secondary storage devices. To remove the influences from the multi-user operating system (caching, network access,...) we repeated all experiments 5 times. Fig. 2 shows the average results for the ES-mining problem, for different values of ρ_g and ρ_s. The bottom line shows the time spent on preprocessing and labelling (phases 1 and 2). The top three lines are the total running times (phases 1–3) for $\rho_g = 6/3, 5/3$ and $4/3$, respectively. As expected, larger values for ρ_g result in shorter running times, because less strings have to be returned. This is also the case for larger values of ρ_s.

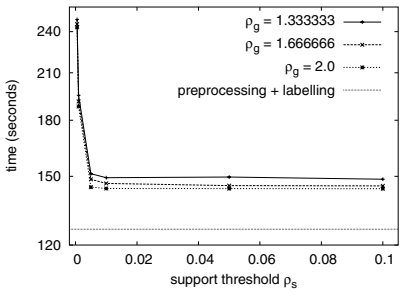


Fig. 2. Times for mining emerging substrings in two databases of appr. 23MB each. The bottom line shows the time for preprocessing. The top three lines represent different choices of ρ_g.

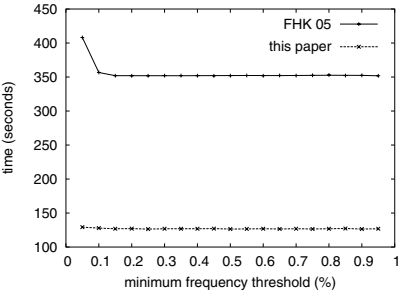


Fig. 3. Times for mining frequent patterns in the same two databases as in Fig. 2, compared with our older method FHK'05 [2]. The maximum frequency threshold was held fixed at 95%.

Fig. 3 shows results for mining frequent substrings. Again, we used the human dataset as the positive database, and the mouse dataset as the negative one. The maximum frequency threshold for MOUSE was held fixed at 95%, but similar graphs could be shown for other values. Because we have already shown in [2] that methods based on suffix tries [10,11] are not competitive with suffix-array based methods, we just compared with our previous approach [2]. As one can see in the figure, our new method is 2–3 times faster than our old method. This shows again that the method presented in this paper is also of practical value.

5 Conclusion

We presented a theoretically optimal solution to string mining under frequency constraints. As in previous work, we build upon results on index structures for strings. One of the building blocks is the fast computation of range minimum queries. Given this algorithmic framework, it is possible to compute solutions, e.g., for emerging substrings and patterns statistically associated with classes of sequences, very efficiently. In future work, we will focus on applications, such as finding new binding motifs, consider the integration of syntactic constraints and study the use of persistent index structures for strings.

References

1. Birzele, F., Kramer, S.: A new representation for protein secondary structure prediction based on frequent patterns. Submitted to Bioinformatics
2. Fischer, J., Kramer, S., Heun, V.: Fast frequent string mining using suffix arrays. In: Proc. ICDM, IEEE Computer Society (2005) 609–612
3. Gonnet, G.H., Baeza-Yates, R.A., Snider, T.: New indices for text: PAT trees and PAT arrays. In Frakes, W.B., Baeza-Yates, R.A., eds.: Information Retrieval: Data Structures and Algorithms. Prentice-Hall (1992) 66–82

4. Manber, U., Myers, E.W.: Suffix arrays: A new method for on-line string searches. SIAM J. Comput. **22**(5) (1993) 935–948
5. Kasai, T., Lee, G., Arimura, H., Arikawa, S., Park, K.: Linear-time longest-common-prefix computation in suffix arrays and its applications. In: Proc. CPM. Volume 2089 of LNCS. Springer (2001) 181–192
6. Berkman, O., Vishkin, U.: Recursive star-tree parallel data structure. SIAM J. Comput. **22**(2) (1993) 221–242
7. Fischer, J., Heun, V.: Theoretical and practical improvements on the RMQ-problem, with applications to LCA and LCE. In: Proc. CPM. Volume 4009 of LNCS. Springer (2006) 36–48
8. Wang, L., Zhao, H., Dong, G., Li, J.: On the complexity of finding emerging patterns. In: Proc. COMPSAC - Workshops and Fast Abstracts, IEEE Press (2004) 126–129
9. Chan, S., Kao, B., Yip, C.L., Tang, M.: Mining emerging substrings. In: Proc. DASFAA, IEEE Computer Society (2003) 119–126
10. De Raedt, L., Jäger, M., Lee, S.D., Mannila, H.: A theory of inductive query answering. In: Proc. ICDM, IEEE Computer Society (2002) 123–130
11. Lee, S.D., De Raedt, L.: An efficient algorithm for mining string databases under constraints. In: Proc. KDID 2004. Volume 3377 of LNCS. Springer (2005) 108–129
12. Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. In: Proc. KDD, ACM Press (1999) 43–52
13. Ko, P., Aluru, S.: Space efficient linear time construction of suffix arrays. In: Proc. CPM. Volume 2676 of LNCS. Springer (2003) 200–210
14. Schürmann, K.B., Stoye, J.: An incomplex algorithm for fast suffix array construction. In: Proceedings of ALENEX/ANALCO, SIAM Press (2005) 77–85
15. Manzini, G., Ferragina, P.: Engineering a lightweight suffix array construction algorithm. Algorithmica **40**(1) (2004) 33–50
16. Manzini, G.: Two space saving tricks for linear time lcp array computation. In: Proc. SWAT. Volume 3111 of LNCS. Springer (2004) 372–383
17. Arlazarov, V.L., Dinic, E.A., Kronrod, M.A., Faradzev, I.A.: On economic construction of the transitive closure of a directed graph. Dokl. Acad. Nauk. SSSR **194** (1970 (in Russian)) 487–488 English translation in *Soviet Math. Dokl.*, 11: 1209-1210, 1975.
18. Hui, L.C.K.: Color set size problem with application to string matching. In: Proc. CPM. Volume 644 of LNCS. Springer (1992) 230–243
19. Gusfield, D.: Algorithms on Strings, Trees, and Sequences. Cambridge University Press (1997)
20. Abouelhoda, M.I., Kurtz, S., Ohlebusch, E.: Replacing suffix trees with enhanced suffix arrays. J. Discrete Algorithms **2**(1) (2004) 53–86
21. Phan, I.Q.H., Pilbout, S.F., Fleischmann, W., Bairoch, A.: NEWT, a new taxonomy portal. Nucleic Acids Res **31**(13) (2003) 3822–3823

k-Anonymous Decision Tree Induction

Arik Friedman, Assaf Schuster, and Ran Wolff

Technion – Israel Institute of Technology
Computer Science Dept.
{arikf, assaf, ranw}@cs.technion.ac.il

Abstract. In this paper we explore an approach to privacy preserving data mining that relies on the k-anonymity model. The k-anonymity model guarantees that no private information in a table can be linked to a group of less than k individuals. We suggest extended definitions of k-anonymity that allow the k-anonymity of a data mining model to be determined. Using these definitions, we present decision tree induction algorithms that are guaranteed to maintain k-anonymity of the learning examples. Experiments show that embedding anonymization within the decision tree induction process provides better accuracy than anonymizing the data first and inducing the tree later.

Keywords: k-anonymity, privacy preserving data mining, decision trees.

1 Introduction

In recent years, the effectiveness of data mining tools in revealing the knowledge locked within huge databases has raised concerns about its impact on the privacy of individuals. Two main approaches to privacy preserving data mining were suggested. The *data transformation approach* – e.g., [1] – tries to modify the data so as to hide the sensitive information while retaining interesting patterns. The *cryptographic approach* – e.g., [2,3,4] – uses cryptographic tools to prevent information leakage during the computation of the data mining model. The latter approach only applies to distributed data mining and does not prevent leakage due to the model itself (see, for example, [5]).

k-anonymity – a definition for privacy that was conceived in the context of databases – has come a long way in the public arena. Roughly speaking, k-anonymity provides a "blend into the crowd" approach to privacy. It assumes that the owner of a data table can separate the columns into *public* ones (*quasi-identifiers*) and *private* ones. Public columns may appear in external tables, and thus be available to an attacker. Private columns contain data which is not available in external tables and needs to be protected. The guarantee provided by k-anonymity is that an attacker would not be able to link private information to groups of less than k individuals. This is enforced by making certain that every combination of public attribute values appears in at least k rows of the released table, or in no row at all. k-anonymity is accepted today by both legislators and corporations, and is considered to provide the kind of privacy required by legislation such as the HIPAA [6].

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 151–162, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Table anonymization is NP-Hard [7]. Thus, heuristic efficient anonymization of tables is the concern of most work in the area [8,9,10,11,12,13]. Specific care is given to preserving as much of the original data as possible. Interestingly, some of this work deals with preserving data which would be useful should the table be data mined following its release [9,10,11]. Data mining is envisioned as the main target application of released data.

This paper takes a direct approach for the combination of k-anonymity and data mining. Rather than asking how data can be anonymized so that it is useful for data mining, we ask how can data be mined so that the resulting model is guaranteed to provide k-anonymity. We specifically discuss this question in the context of decision tree induction. In this context, anonymity is at risk when the decision tree overfits a small set of learning examples and allows the attacker to predict their private attribute values. We describe a decision tree induction algorithm whose output is guaranteed not to compromise the k-anonymity of the data from which it was induced. An independent work [14] presents a similar concept in the context of itemset mining. However, that work does not differentiate public attributes from private attributes, and is limited to binary attributes. In addition, anonymity is achieved by postprocessing of the data mining output, while we suggest an integration of the two processes.

Our approach is superior to existing methods (e.g., [9,10,11]), which guarantee k-anonymity of a data mining model by building it from a k-anonymized table. For the sake of efficiency, these methods generalize attributes homogenously over all the tuples. This kind of anonymization was termed *single-dimension recoding* [15]. Using our method, however, attributes can be generalized differently in each tuple, depending on other attribute values. This kind of anonymization was termed *multi-dimensional recoding*. Furthermore, in the existing methods, heuristic cost metrics are the driving force. For example, a classification metric, which is in essence the classification error over the entire data, may be used. Such metrics are not necessarily optimal for a specific data mining task. We show that a decision tree induced using our method is usually more accurate than that induced by existing methods. Needless to say, both decision trees provide the same level of anonymity.

This paper makes the following contributions:

- It suggests extended definitions of k-anonymity, which allow the k-anonymity of a data mining model with respect to the learning examples to be determined.
- It demonstrates how the definitions of k-anonymity can be applied to determine the anonymity of a decision tree.
- It presents a decision tree induction algorithm which guarantees k-anonymous output and which performs better than existing methods in terms of accuracy on standard benchmarks.

The organization of this paper is as follows: Section 2 outlines the extended definitions of k-anonymity of data mining models. Section 3 demonstrates

how the definitions can be incorporated within decision tree induction algorithms to guarantee k-anonymous output. Section 4 compares this new algorithm experimentally to previous work. Conclusions are drawn in Sect. 5.

2 Extending k-Anonymity to Models

We start by extending the definition of k-anonymity beyond the release of tables. Just as in the original k-anonymity model, we assume that the data owner can determine which attributes are known to a potential attacker and can be used to identify individuals, and which attributes are private knowledge. Additionally, without loss of generality, we assume that each tuple in the database pertains to a different individual.

Definition 1 (A Private Database). *A private database T is a collection of tuples from a domain $D = A \times B = A_1 \times ... \times A_k \times B_1 \times ... \times B_\ell$. $A_1, \ldots, A_k$ are public attributes (a.k.a. quasi-identifiers) and $B_1, \ldots, B_\ell$ are private attributes.*

We denote $A = A_1 \times \ldots \times A_k$ the *public subdomain* of D. For every tuple $x \in D$, the projection of x into A, denoted x_A, is the tuple in A that has the same assignment to each public attribute as x. The projection of a table T into A is denoted $T_A = \{x_A : x \in T\}$.

Definition 2 (A Model). *A model M is a function from a domain D to an arbitrary output domain O.*

Every model induces an equivalence relation on D, i.e., $\forall x, y \in D$, $x \equiv y \Leftrightarrow M(x) = M(y)$. The model partitions D into respective equivalence classes such that $[x] = \{y \in D : y \equiv x\}$. The model alone imposes some structure *on the domain*. However, when a data owner releases a model based on a database, it also provides information about how the model relates to the database.

Definition 3 (A Release). *Given a database T and a model M, a release M_T is the pair (M, p_T), where p_T (for population) is a function that assigns to each equivalence class induced by M the number of tuples from T that belong to it, i.e., $p_T([x]) = |T \bigcap [x]|$.*

In our terminology, a decision tree model is a function that assigns bins to tuples in D. Accordingly, every bin within every leaf constitutes an equivalence class. Two tuples which fit into the same bin cannot be distinguished from one another using the tree, even if they do not agree on all attribute values. A release of a decision tree includes the partition into bins, as well as the number of learning examples that populate each bin.

Note that other definitions of a release, in which the kind of information provided by p_T is different, are possible as well. For example, a decision tree may provide the relative frequency of a bin within a leaf, or just denote the bin that constitutes the majority class. In this paper we assume the worst case, in which the exact number of learning examples in each bin is provided. The effect

of different kinds of release functions on the extent of private data that can be inferred by an attacker is an open question. Nevertheless, the anonymity analysis provided herein can be applied in the same manner. In other words, different definitions of p_T would reveal different private information on the same groups of tuples.

We now turn to see how the database and the model are perceived by an attacker. One of the fundamental assumptions of the k-anonymity model is about the data available to the attacker.

Definition 4 (A Public Identifiable Database). *A public identifiable database $T_{\mathrm{ID}} = \{(\mathrm{id}_x, x_A) : x \in T\}$ is a projection of a private database T into the public subdomain A, such that every tuple of T_A is associated with the identity of the individual to whom the original tuple in T pertained.*

Although the attacker knows only the values of public attributes, he can nevertheless try to use the release M_T to expose private information of individuals represented in T_{ID}. Consider a tuple $(\mathrm{id}_x, x_A) \in T_{\mathrm{ID}}$. As each equivalence class in M may rely on both private and public attributes, the attacker, could he associate x_A with the correct equivalence class, could then infer which private attribute values are possible for x according to this equivalence class. However, depending on the unknown private attribute values of x, there might be a number of possible equivalence classes $[x]$ to which an attacker can associate x_A. We call this set of equivalence classes the *span* of x_A.

Definition 5 (A Span). *Given a model M, the span of a tuple $a \in A$ is the set of equivalence classes induced by M and which contain tuples $x \in D$, whose projection into A is a. Formally, $S_M(a) = \{[x] : x \in D \wedge x_A = a\}$. When M is evident from the context, we will use the notation $S(a)$.*

For example, in a decision tree, the span of a tuple is the set of bins to which the tuple can be routed when any combination of private attribute values is possible for it. Given a public identifiable database T_{ID} and a model M, we use $S(a)_{T_{\mathrm{ID}}} = \{(\mathrm{id}_x, x_A) \in T_{\mathrm{ID}} : S(x_A) = S(a)]\}$ to denote the set of tuples that appear in T_{ID} and whose span is $S(a)$. These are tuples from T_{ID} which are indistinguishable with respect to the model M – each of them is associated with the same set of equivalence classes of M. Just as associating an individual with an equivalence class would allow private attribute values to be inferred, knowing the values of p_T for each equivalence class in $S(a)$ allows the possible private attribute value combinations for the tuples in $S(a)_{T_{\mathrm{ID}}}$ to be constrained, hence compromising the privacy of the individuals.

Definition 6 (Linking Attack Using a Release). *A linking attack using a release M_T and a public identifiable database T_{ID} is performed by grouping tuples in T_{ID} according to their spans. Each group of tuples is then linked to the list of possible private attribute value combinations, according to M_T.*

Definition 7 (k-Anonymous Model). *A model M is k-anonymous with respect to a private table T if a linking attack on the tuples in T_{ID} using the release*

M_T will not succeed in linking private data to fewer than k individuals. In other words, a model M is k-anonymous with respect to T if, for every $(\mathrm{id}_x, x_A) \in T_{\mathrm{ID}}$, $|S(x_A)_{T_{\mathrm{ID}}}| \geq k$.

For example, consider the decision tree in Fig. 1. The decision tree was formed using the data in Table 1. The *Marital Status* attribute is public, while the *Sports Car* and *Loan Risk* attributes are private. The *Loan Risk* attribute is the class attribute. The decision tree includes six bins, each with its population denoted in the tree. The tuples of *Anna* and *Ben* belong to the same bin, because of the *Sports Car* and *Loan Risk* attributes. The model ignores the value of the *Marital Status* attribute for these tuples. On the other hand, an attacker, who has no access to the *Sports Car* attribute values, is forced to consider routing *Anna*'s tuple to the leaf $l_{\mathrm{Unmarried}}$, and routing *Ben*'s tuple to the leaf l_{Married}, in addition to routing each of them to the leaf l_{No}. Therefore, the decision tree implies two spans: $\{l_{\mathrm{Married/good}}, l_{\mathrm{Married/bad}}, l_{\mathrm{no/good}}, l_{\mathrm{no/bad}}\}$ for *John*, *Ben*, and *Laura*, and $\{l_{\mathrm{Unmarried/good}}, l_{\mathrm{Unmarried/bad}}, l_{\mathrm{no/good}}, l_{\mathrm{no/bad}}\}$ for *Lisa*, *Robert*, and *Anna*. The attacker can use the populations to induce the distribution of class attribute values within each span, but he cannot know, for example, which of the three tuples in the first span belongs to $l_{\mathrm{Married/good}}$. As each span contains three tuples, the model is 3-anonymous with respect to Table 1.

Table 1. Mortgage company data

Name	Marital Status	*Sports* Car	*Loan Risk*
Lisa	Unmarried	*Yes*	*good*
John	Married	*Yes*	*good*
Ben	Married	*No*	*bad*
Laura	Married	*No*	*bad*
Robert	Unmarried	*Yes*	*bad*
Anna	Unmarried	*No*	*bad*

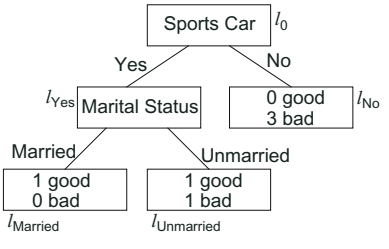


Fig. 1. A 3-anonymous decision tree

3 Inducing *k*-Anonymous Decision Trees

This section presents an algorithm which induces k-anonymous decision trees. The algorithm is based on the well-known ID3 algorithm [16] and on its extension, C4.5 [17]. ID3 applies greedy hill-climbing to construct a decision tree. Starting from a root that holds the entire learning set, it chooses the attribute that maximizes the information gain, and splits the current node into several new nodes. The learning set is then divided among the new nodes according to the value each tuple takes on the chosen attribute, and the algorithm is applied recursively on the new nodes.

The k-anonymity preserving equivalent of ID3, kADET (Algorithm 3.1), uses the same hill-climbing approach, with two changes: First, when considering

all possible splits of a node, kADET eliminates splits which would lead to k-anonymity breach. Second, the algorithm is not recursive. Instead, all the potential splits are considered in a single priority queue and the best one of all those that retain k-anonymity is picked. This method is required since k-anonymity is defined in terms of spans, which may include bins from several decision tree nodes. A decision regarding one node may thus influence other nodes.

3.1 *k*ADET Algorithm

The input of kADET is a private database T, the public attributes P, the private attributes Q, the class attribute C, and the anonymity parameter k. First, kADET computes the initial set of equivalence classes (bins) and spans: a single span containing all of the bins and all of the tuples if the class is private, and as many spans as bins, with each span containing a single bin and its tuple population if the class is public. If one of the spans contains less than k tuples from T, kADET returns *nil* and terminates. Otherwise, kADET creates the initial queue of possible splits, where each candidate split contains the root node and an attribute from P or Q. The queue is ordered according to the gain from each split. kADET then enters its main loop.

The main loop of kADET has the following steps: First, the most gainful candidate split $(node, attribute, gain)$ is popped out of the queue. If the node regarded in this candidate is already split, the candidate is purged. Otherwise, kADET tests whether splitting the node according to the suggested attribute would breach k-anonymity. If it would, then, again, this candidate is purged. However, if the attribute can be generalized, then a new candidate is inserted to the queue, this time with the generalized attribute. Finally, if k-anonymity is not breached, the node is split.

Several actions are taken in the splitting of a node: First, every bin of the parent node is split between the descendant nodes, according to the value of the splitting attribute. Accordingly, every span that contains this bin is updated with the new list of bins. The descendant nodes inherit the list of spans from their parent, and are added to the lists of nodes of those spans. If the splitting attribute is private, no further action is required, as the attacker cannot distinguish between the new bins. However, if the splitting attribute is public, then the attacker can use the split to distinguish tuples. Specifically, tuples that are routed to one of the new nodes will not be routed to its sibling nodes. Therefore, each of the spans in the split node, is split into new spans, one for each new descendant node. Each new span contains the bins from the original span, except for those of the sibling nodes. Likewise, the population of the original span is divided according to the value the tuples take on the splitting attribute. Nodes whose bins are contained in the new spans, and which are not descendant of the original node, are associated with all of the new spans.

Figure 2 demonstrates an execution of *kADET*, using the data in Table 1 as input. *Marital Status* is a public attribute; *Sports Car* and *Loan Risk* are private attributes. The result of the execution is the decision tree in Fig. 1.

Algorithm 1. k-Anonymous Decision Tree (kADET)

Input: T – private dataset, P – public attributes, Q – private attributes, C – the class attribute, k – anonymity parameter
procedure MAIN
  Create $root$ node in $Tree$
  Create in $root$ one bin b_c for each value $c \in C$ and divide T among the bins
  if $C \in P$ **then**
    Create one span S_c for every value $c \in C$,
      $S_c.Bins \leftarrow \{b_c\}$, $S_c.Population \leftarrow b_c.Population$, $S_c.Nodes \leftarrow \{root\}$
    set $root.Spans$ to the list of all spans
  else
    Create a single span s. Set $s.Bins$ to the list of all bins,
      $s.Population \leftarrow T$, $s.Nodes \leftarrow \{root\}$, $root.Spans \leftarrow \{s\}$
  if $\exists span \in root.Spans$ such that $0 < |span.Population| < k$ **then return** nil
  for $att \in P \bigcup Q \setminus \{C\}$ **do** add $(root, att, \text{gain}(root, att))$ to $Queue$
  while $Queue$ has elements with positive gain **do**
    Let $(n, a, gain) = \arg\max_{gain} \{Queue\}$
    if $n.sons \neq \emptyset$ **then continue**
    if $Breach\,(n, a, k)$ **then**
      if a has generalization a **then** insert $(n, a\,, \text{gain}(n, a\,))$ to $Queue$
    else Split(n, a)
  Set the Class variable in each leaf to the value with the largest bin.
  return $Tree$

---

procedure BREACH($node$, att, k)
  if $att \in Q$ **then return** $false$
  for $v \in att.values$ and $span \in node.Spans$ **do**
    if $0 < |\{t \in span.Population : t[att] = v\}| < k$ **then return** $true$
  return $false$

---

procedure SPLIT($node$, att)
  for $v \in att.values$ **do**
    Let $node.sons[v]$ be a new descendant node
    Let $node.sons[v].Bins[b]$ be a new bin, which refines $node.Bins[b]$ such that
      $node.sons[v].Bins[b].tuples \leftarrow \{t \in node.Bins[b].tuples : t[att] = v\}$
    Let $node.sons[v].Spans \leftarrow node.Spans$
  for $span \in node.Spans$ **do** replace each bin of the original node with its refinements, computed above, and add the new nodes to $span.Nodes$
  if $att \in P$ **then**
    for $span \in node.Spans$ **do**
      Remove $span$ from every node $n \in span.Nodes$
      for $v \in att.values$ **do**
        Create a new span s_v
        $s_v.Nodes \leftarrow span.Nodes \setminus \{node.sons[u] : u \neq v\}$
        $s_v.Bins \leftarrow span.Bins \setminus \{bin \in node.sons[u].Bins : u \neq v\}$
        $s_v.Population \leftarrow \{t \in span.Population : t[att] = v\}$
        Add s to $node.sons[v].spans$
        Add s to every node $n \in span.Nodes \setminus node.sons$

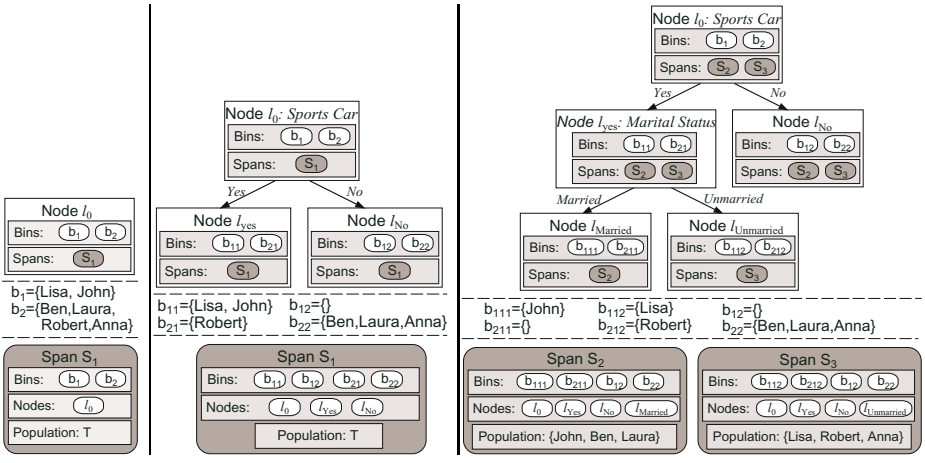


Fig. 2. Execution of *kADET*

3.2 Correctness and Overhead Analysis

The key to proving correctness of the algorithm is in showing that the population of each span, as computed by the algorithm, is the same as the one defined in Sect. 2: the set of tuples which, without knowledge of private attribute values, can be routed to the same set of bins. The proof is omitted due to lack of space.

The computational overhead incurred by the algorithm, respective to that of ID3, stems from the need to compute and track all of the different spans. In the worst case, the number of spans may reach the size of the public domain A. To see how, consider a tree in which all of the top nodes split on private attributes, until the number of leaves is equal to the number of public attributes and then, every leaf is split according to a *different* public attribute. The number of spans in the tree is equal to the size of A.

While this overhead is indeed high, it is inherent to the problem, because of the need to validate that every span is populated by k tuples or more. In practice, the number of spans will be much smaller. For example, when only the class attribute is private, the number of spans is the number of leaves.

3.3 From ID3 to C4.5

C4.5 was introduced by Quinlan [17] in order to extend and improve ID3. It implements better attribute scoring metrics (gain ratio instead of gain), error-based pruning, continuous attribute quantization, and treatment of missing values. All these extensions, other than the change of the scoring function – which has no effect on privacy – require careful analysis when used to extend kADET.

Pruning. C4.5 uses error-based pruning in two ways: discarding subtrees and replacing them with leaves, and replacing subtrees with one of their branches.

Using the first method is safe – undoing a split unifies equivalence classes, and may unify spans, meaning that the population of a span can only increase. The second method, however, may cause a k-anonymous tree to become non-k-anonymous, as it induces different spans with different populations. Therefore we avoid this technique in our implementation.

Continuous attributes. In the C4.5 algorithm, continuous attributes are handled by creating binary splits. The algorithm considers all the possible split points, and chooses the one with the best information gain. We implemented the same approach, adding the constraint that a split point should not cause a breach of k-anonymity.

Missing values. Missing values extend the k-anonymity model in ways which have not been modelled yet. It is not clear, for instance, whether a value that is missing in the learning examples would be missing in the data available to the attacker. We leave the extension of the k-anonymity model to missing values for further research.

4 Evaluation

To conduct our experiments we implemented the algorithms using the Weka package [18]. We use as a benchmark the Adults database from the UC Irvine Machine Learning Repository [19], which contains census data, and has become a commonly used benchmark for k-anonymity. The data set has 6 continuous attributes and 8 categorial attributes. The class attribute is income level, with two possible values, $\leq$50K or >50K. After records with missing values have been removed, there are 30,162 records for training and 15,060 records for testing (of which 24.5% are classified >50K). For the categorial attributes we use the same generalization hierarchies described in [10]. For the ID3 experiments we dropped the continuous attributes, because of ID3 limitations. The experiment was performed on a 3.0GHz Pentium IV processor with 512MB memory.

The anonymized decision trees algorithms use the training data to induce an anonymous decision tree. Then the test data (in a non-anonymized form) is classified using the anonymized tree. For all values of k the decision tree induction took less than 4 seconds for ID3, and less than 10 seconds for C4.5.

4.1 Accuracy vs. Anonymity Tradeoffs

Our first goal is to assess the tradeoff between classification accuracy and the privacy constraint.

Figure 3 shows the classification error of the anonymous ID3 for various k parameters, compared to the classification error for ID3 and C4.5. In spite of the anonymity constraint, the classifier maintains good accuracy. At $k = 750$ there is a local optimum when the root node is split using the *Relationship* attribute. At $k = 1000$ this attribute is discarded because of an anonymity breach, and the *Marital Status* attribute is chosen instead, yielding better classification.

We compared our results with those obtained using the top-down specialization (TDS) algorithm presented in [10], the goal of which is to produce

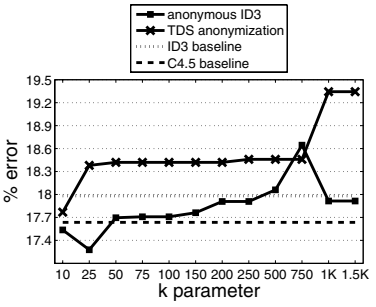


Fig. 3. Classification error vs. k parameter for ID3

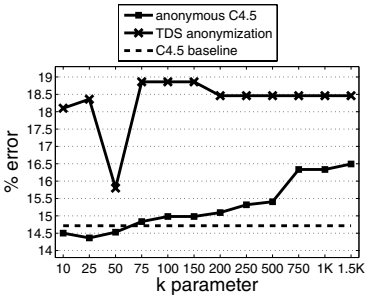


Fig. 4. Classification error vs. k parameter for C4.5

anonymized data useful for classification problems. The algorithm starts with the topmost generalization level, and iteratively chooses attributes to specialize, using a metric that measures the information gain for each unit of anonymity loss. The same generalization scheme is applied on all the tuples. We note that TDS uses both training and test data to choose a generalization. This may provide different generalization results, though not necessarily better or worse than those obtained when generalizing the training data alone. TDS results also appear in Fig. 3. In contrast to the TDS algorithm, our algorithm can apply different generalizations on different groups of tuples, and it achieves an average reduction of 0.6% in classification error with respect to TDS.

Figure 4 shows a similar comparison using all 14 attributes of the Adult dataset with the anonymous C4.5 algorithm. The large size of the quasi-identifier affects the accuracy of the TDS generalization, and our algorithm reduces the classification error by an average of 3% with respect to TDS.

4.2 Privacy Risks and ℓ-Diversity

k-anonymity makes no restriction regarding private attribute values. Therefore, it is possible that a k-anonymous model would allow a complete inference of these values. In this section, our goal is to assess how many individuals are prone to immediate inference attacks and show how such attacks can be thwarted.

We look at the number of individuals for whom an attacker may infer the class attribute value with full certainty. This is the number of tuples associated with spans for which all the tuples share the same class. Because of space limits, we provide only the main figures. For the anonymous ID3, this number drops to zero only for values of k beyond 750, and even then the attacker may still be able to infer attribute values with high probability. The inference problem is less acute in the case of the anonymous C4.5, because of pruning. The number of exposed tuples drops to zero at $k = 75$, and is very low (below 0.5%) even for smaller values of k.

The ℓ-diversity model [20] suggests solving the inference problem by requiring a certain level of diversity in class values for every group of identifiable tuples.

For example, *entropy ℓ-diversity* is maintained when the entropy of the class values for every such group exceeds a threshold value $log(\ell)$.

We altered our algorithms by replacing the $Breach()$ function with one that checks the entropy ℓ-diversity constraint, ruling out splits that violate this constraint. Note that the parameters for k-anonymity and ℓ-diversity are not comparable. In particular, as there are only two class values, the best we can hope for is entropy 2-diversity. This is achieved when there is equal chance for each class value. However, for $\ell < 2$, entropy ℓ-diversity limits the attacker's confidence in inference attacks. The *confidence limit* is the maximal probability of any private value for any individual. The data owner can control the confidence limit by manipulating the ℓ parameter. For example, to deny the attacker the ability to infer a class value with confidence greater than 85%, entropy higher than $0.85 \times \log(1/0.85) + 0.15 \times \log(1/0.15) = 0.61$ should be maintained. This amounts to applying entropy ℓ-diversity with $\ell = 1.526$ ($\log 1.526 = 0.61$).

Following this discussion, Figures 5 and 6 display the tradeoff between the confidence limit and the accuracy of the induced decision trees. So long as the confidence threshold is high enough, it is possible to induce decision trees without a significant accuracy penalty. The lowest achievable confidence level is 75.1%, as it pertains to the class distribution in the root node. In the case of ID3, every split of the root node results in a node with confidence greater than 85%. Therefore, a confidence limit of 85% or lower prohibits the induction of a useful decision tree. The additional attributes available to the C4.5 algorithm allow the boundary to be stretched to a lower confidence threshold.

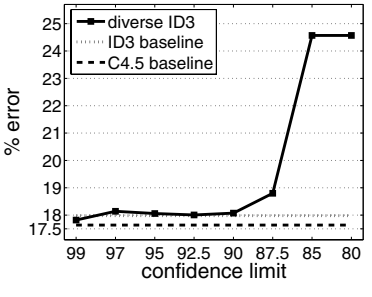


Fig. 5. Confidence level vs. error rate for ID3, 9 attributes

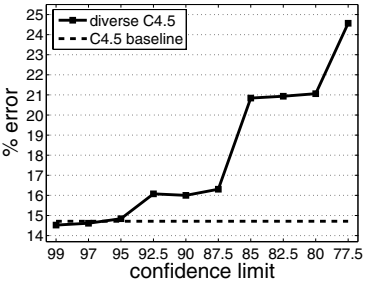


Fig. 6. Confidence level vs. error rate for C4.5, 15 attributes

5 Conclusions

In this paper we presented decision tree induction algorithms which guarantee k-anonymous output. Using our definitions, it is possible to introduce similar constraints in other data mining algorithms as well. Another interesting use of this method is promoted by the ability to construct a table that is equivalent to a data mining model. Such a table would maintain k-anonymity, while preserving

the data patterns evident in the data mining model. Hence data mining algorithms can be used as templates for pattern-preserving anonymization schemes.

References

1. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proc. of the ACM SIGMOD Conference on Management of Data, ACM Press (2000) 439–450
2. Du, W., Zhan, Z.: Building decision tree classifier on private data. In: Proc. of CRPITS'14, Darlinghurst, Australia, Australian Computer Society, Inc. (2002) 1–8
3. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Proc. of CRYPTO '00, London, UK, Springer-Verlag (2000) 36–54
4. Vaidya, J., Clifton, C.: Privacy-preserving decision trees over vertically partitioned data. In: DBSec. (2005) 139–152
5. Kantarcioğlu, M., Jin, J., Clifton, C.: When do data mining results violate privacy? In: Proc. of ACM SIGKDD, NY, USA, ACM Press (2004) 599–604
6. US Dept. of HHS: Standards for privacy of individually identifiable health information; final rule (2002)
7. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: Proc. of PODS '04, New York, NY, USA, ACM Press (2004) 223–228
8. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Approximation algorithms for k-anonymity. In: Journal of Privacy Technology (JOPT). (2005)
9. Bayardo Jr., R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proc. of ICDE. (2005) 217–228
10. Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: Proc. of ICDE. (2005)
11. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: Proc. of ACM SIGKDD. (2002) 279–288
12. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proc. of ICDE. (2006)
13. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems **10**(5) (2002) 571–588
14. Atzori, M., Bonchi, F., Giannotti, F., Pedreschi, D.: k-anonymous patterns. In: Proc. of PKDD. (2005) 10–21
15. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proc. of SIGMOD, NY, USA, ACM Press (2005) 49–60
16. Quinlan, J.R.: Induction of decision trees. Machine Learning **1**(1) (1986) 81–106
17. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
18. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2nd edn. Morgan Kaufmann, San Francisco (2005)
19. D.J. Newman, S. Hettich, C.B., Merz, C.: UCI repository of machine learning databases (1998)
20. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ-diversity: Privacy beyond k-anonymity. In: Proc. of ICDE. (2006)

Closed Sets for Labeled Data*

Gemma C. Garriga[1], Petra Kralj[2], and Nada Lavrač[2,3]

[1] Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034 Barcelona, Spain
garriga@lsi.upc.edu
[2] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
[3] University of Nova Gorica, Vipavska 13, 5000 Nova Gorica, Slovenia
petra.kralj@gmail.com
nada.lavrac@ijs.si

Abstract. Closed sets are being successfully applied in the context of compacted data representation for association rule learning. However, their use is mainly descriptive. This paper shows that, when considering labeled data, closed sets can be adapted for prediction and discrimination purposes by conveniently contrasting covering properties on positive and negative examples. We formally justify that these sets characterize the space of relevant combinations of features for discriminating the target class. In practice, identifying relevant/irrelevant combinations of features through closed sets is useful in many applications. Here we apply it to compacting emerging patterns and essential rules and to learn descriptions for subgroup discovery.

1 Introduction

Rule discovery has been addressed from two different perspectives: data mining and machine learning. Data mining mainly explores unlabeled data, and the focus resides on finding all rules over a certain confidence that summarize the original data. On the other hand, machine learning is mainly concerned with the analysis of class labeled data, resulting in the induction of classification and prediction rules, and—more recently—also descriptive rules that aim at providing insightful knowledge from the data (subgroup discovery, contrast set mining). Traditional rule learning algorithms for classification include CN2 [3] and Ripper [4]. Other approaches have been proposed that are based on the association rule technology but are applied to class labeled data (e.g., the Apriori-C classifier [8] and the Essence algorithm for inducing "essential" classification rules based on the covering properties of frequent itemsets [1]).

In subgroup discovery the aim is to find subgroup descriptions that are characteristic for examples with a certain property of interest, and the closely related contrast set mining aims at capturing discriminating features that contrast instances between classes. Special rule learning algorithms for subgroup discovery include Apriori-SD [9], CN2-SD [11] or SD [7]. These descriptive mining algorithms aim at finding characteristic rules as combinations of features with high

* This work has been partly supported by the PASCAL Network of Excellence.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 163–174, 2006.
© Springer-Verlag Berlin Heidelberg 2006

coverage. If there are several rules with the same coverage, most specific rules (with more features) are appropriate for description and explanation purposes. On the other hand, algorithms for contrast set mining are STUCCO [2], and recently, an innovative approach was presented in the form of mining Emerging Patterns [5]. Basically, Emerging Patterns (EP) are sets of features in the data whose supports change significantly from one class to another.

Indeed, we can see all these described tasks on labeled data (learning classification rules, subgroup discovery, or contrast set mining) as a process of searching a space of concept descriptions (hypotheses in the form of rule antecedents). Some descriptions in this hypothesis space may turn out to be more relevant than others for characterizing and/or discriminating the target class. Searching for relevant descriptions for rule construction has been extensively addressed in descriptive data mining. A useful insight was provided by closure systems, aimed at compacting the whole space of descriptions into a reduced system of relevant sets that formally conveys the same information as the complete space. The approach has successfully evolved towards mining *closed itemsets* (see e.g. [12,14]). Intuitively, closed itemsets can be seen as maximal sets of items/features covering a maximal set of examples. Despite its success in the data mining community, the use of closed sets is mainly descriptive. For example, they can be used to limit the number of association rules produced without information loss.

To the best of our knowledge, the notion of closed sets has not yet been exported to labeled data, neither used in the learning tasks for labeled data described above. In this paper we show that raw closed sets can be adapted for discriminative purposes by conveniently contrasting covering properties on positive and negative examples. Moreover, thanks to the final structural properties and the feature filtering theory of [10], we formally justify that our obtained sets characterize the space of relevant combinations of features for discriminating the target class.

In practice, our approach to discovering closed sets from labeled data, (described in Sections 3 and 4) turns out to be very useful in many applications: from constructing rule based classifiers of increased accuracy, to finding most interpretative descriptions for subgroup discovery, among others. In particular, we have applied our proposal to reduce the number of EPs and to compress the number of essential rules (Section 6.1), and finally, to learn descriptions for subgroup discovery (Section 6.2).

2 Background

Features, used for describing the training examples, are logical variables representing attribute-value pairs (called items in association rule learning). If $F = \{f_1, \ldots, f_n\}$ is a fixed set of features, we can represent a training example as a tuple of features $f \in F$ with an associated class label. For instance, Table 1 contains examples for the simplified problem of contact lens prescriptions [13]. Patients are described by four attributes and each tuple is labeled with a class label: none, soft or hard. Here F is the set of all attribute-value pairs in the

data, i.e. $F = \{\text{Age=young}, \ldots, \text{Tear=normal}\}$ (the class label is not included in F). This dataset is known to be complete and we will use it throughout the paper to ease the understanding of our proposals.

We consider two-class learning problems where the set of examples E is divided into positives (P, labeled by $+$) and negatives (N, labeled by $-$), and $E = P \cup N$. Multi-class problems can be translated to a series of two-class learning problems. For instance, when the class soft of Table 1 is the target class (in Table 2), all examples labeled with none and hard are considered negative.

Table 1. The contact lens data set

Id	Age	Spectacle prescription	Astig.	Tear prod.	Lens
1	young	myope	no	normal	soft
2	young	hypermetrope	no	normal	soft
3	pre-presbyopic	myope	no	normal	soft
4	pre-presbyopic	hypermetrope	no	normal	soft
5	presbyopic	hypermetrope	no	normal	soft
6	young	myope	no	reduced	none
7	young	myope	yes	reduced	none
8	young	hypermetrope	no	reduced	none
9	young	hypermetrope	yes	reduced	none
10	pre-presbyopic	myope	no	reduced	none
11	pre-presbyopic	myope	yes	reduced	none
12	pre-presbyopic	hypermetrope	no	reduced	none
13	pre-presbyopic	hypermetrope	yes	reduced	none
14	pre-presbyopic	hypermetrope	yes	normal	none
15	presbyopic	myope	no	reduced	none
16	presbyopic	myope	no	normal	none
17	presbyopic	myope	yes	reduced	none
18	presbyopic	hypermetrope	no	reduced	none
19	presbyopic	hypermetrope	yes	reduced	none
20	presbyopic	hypermetrope	yes	normal	none
21	young	myope	yes	normal	hard
22	young	hypermetrope	yes	normal	hard
23	pre-presbyopic	myope	yes	normal	hard
24	presbyopic	myope	yes	normal	hard

Table 2. In this table we show the positive examples when the class soft is selected as the target class (thus, forming the set of examples in P). Instances of the classes none and hard will be considered non-target, thus treated together as negative data N.

Id	Age	Spectacle prescription	Astig.	Tear prod.	Class
1	young	myope	no	normal	+
2	young	hypermetrope	no	normal	+
3	pre-presbyopic	myope	no	normal	+
4	pre-presbyopic	hypermetrope	no	normal	+
5	presbyopic	hypermetrope	no	normal	+

Given a rule $X \rightarrow +$ formed from a set of features $X \subseteq F$: true positives (TP) are those positive examples covered by the rule, i.e. $p \in P$ such that $X \subseteq p$, and false positives (FP) are those negative examples covered by the rule, i.e. $n \in N$ such that $X \subseteq n$; reciprocally, true negatives (TN) are those negative examples not covered by X.

2.1 Relevant Features for Discrimination

The main aim of the theory of relevancy, described in [10] is to reduce the hypothesis space by eliminating irrelevant features from F in the pre-processing phase. As proposed by the authors:

Definition 1 (Coverage of features). *Feature $f \in F$ covers another feature $f' \in F$ if and only if* $\text{TP}(f') \subseteq \text{TP}(f)$ *and* $\text{TN}(f') \subseteq \text{TN}(f)$ *(or equivalently,* $\text{TP}(f') \subseteq \text{TP}(f)$ *and* $\text{FP}(f) \subseteq \text{FP}(f')$*).*

Then, it is stated that $f' \in F$ is *relatively irrelevant* if there exists another feature f such that f covers f'. To illustrate this notion we take the data of Table 1: if examples of class none form our positives and the rest of examples are considered

negative, then the feature Tear=reduced covers Age=young, hence making this last feature irrelevant for the discrimination of the none class.

2.2 Closed Itemsets

From the practical point of view of data mining algorithms, closed itemsets are maximal sets among those other itemsets occurring in the same examples. Formally, let $\text{supp}(X)$ denote the number of examples where the itemset $X \subseteq F$ is contained. Then: a set $X \subseteq F$ is said to be *closed* when there is no other set $Y \subseteq F$ such that $X \subset Y$ and $\text{supp}(X) = \text{supp}(Y)$. In the example from Table 2 the itemset corresponding to {Age=young} is not closed because it can be extended to the maximal set {Age=young, Astigmatism=no, Tear=normal} that has the same support in this data. Notice that by treating positive examples separately, the positive label will be already implicit in the closed itemsets mined on the target class data. Efficient algorithms for discovering closed itemsets over a certain minimum support threshold can be found in [6].

The foundations of closed itemsets are based on the definition of a closure operator on a lattice of items. The standard closure operator Γ for items acts as follows: given a binary relation, the closure $\Gamma(X)$ of a set of items $X \subseteq F$ includes all items that are present in all examples having all items in X. According to the classical theory, operator Γ satisfies the following properties: (1) Monotonicity: $X \subseteq X' \Rightarrow \Gamma(X) \subseteq \Gamma(X')$; (2) Extensivity: $X \subseteq \Gamma(X)$; and (3) Idempotency: $\Gamma(\Gamma(X)) = \Gamma(X)$.

From the formal point of view of Γ, closed sets are those coinciding with their closure, that is, for $X \subseteq F$, X is *closed* iff $\Gamma(X) = X$. Also, when $\Gamma(Y) = X$ for a set $Y \neq X$, it is said that Y is a *generator* of X. By extensitivity of Γ we always have $Y \subseteq X$ for Y generator of X. Closed sets of items can be graphically organized in a Hasse diagram, such as the one depicted in Figure 1 for the closed itemsets mined from data in Table 2.

3 Closed Sets on Target-Class Data

Given an example set $E = P \cup N$ it is trivial to realize that for any rule $X \rightarrow +$ with a set of features $X \subseteq F$, the support of itemset X in P (target class examples) exactly corresponds to the number of true positives of the rule; reciprocally, the support of X in N (non-target class examples) is the number of false positives of the rule. Also, because of the anti-monotonicity property of support (i.e. $Y \subseteq X$ implies $\text{supp}(X) \leq \text{supp}(Y)$) the following useful property can be easily stated. For the sake of simplicity and due to a lack of space, proofs are omitted, although a proof sketch will be provided to justify important results.

Proposition 1. *Let* $X, Y \subseteq F$ *such that* $Y \subseteq X$*, then* $\text{TP}(X) \subseteq \text{TP}(Y)$ *and* $\text{FP}(X) \subseteq \text{FP}(Y)$.

For convenience, let $\text{supp}^+(X)$ denote the support of the set X in the positive set of examples P, and $\text{supp}^-(X)$ the support in the negative set of examples N. Following the last proposition, the next property can be readily seen:

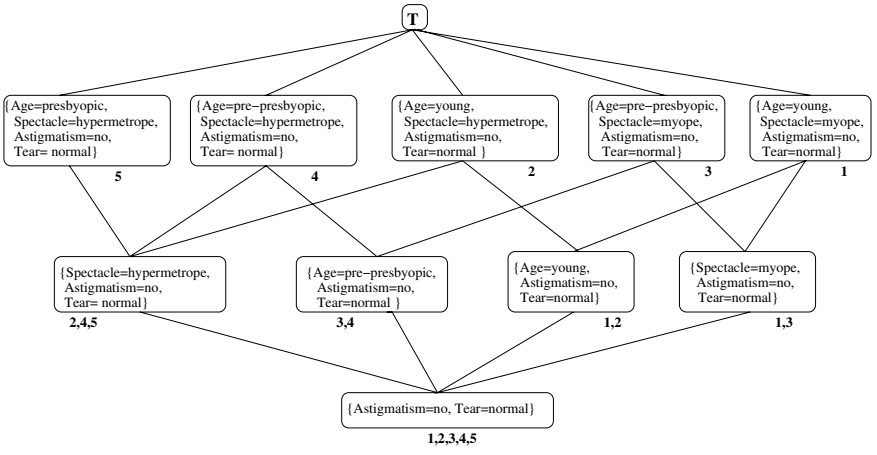


Fig. 1. The lattice of closed itemsets for data in Table 2

Lemma 1. *Feature $f \in F$ covers another feature $f' \in F$ (as in Definition 1), iff* $\text{supp}^+(\{f'\}) = \text{supp}^+(\{f, f'\})$ *and* $\text{supp}^-(\{f\}) = \text{supp}^-(\{f, f'\})$.

Indeed, this last result allows us to rewrite, within the data mining language, the definition of relevancy proposed in [10]: a feature f is *more relevant* than f' when $\text{supp}^+(\{f'\}) = \text{supp}^+(\{f, f'\})$ and $\text{supp}^-(\{f\}) = \text{supp}^-(\{f, f'\})$. In other words, f' is irrelevant with respect to f if the occurrence of f' always implies the presence of f in the positives, and at the same time, f always implies the presence of f' in the negatives.

To the effect of our later arguments it will be useful to cast the result of Lemma 1 in terms of the formal closure operator Γ. Again, because we need to formalize our arguments against positive and negative examples separately, we will use Γ^+ or Γ^- for the closure of itemsets on P or N respectively.

Lemma 2. *A feature f is more relevant than f' iff $\Gamma^+(\{f'\}) = \Gamma^+(\{f, f'\})$ and $\Gamma^-(\{f\}) = \Gamma^-(\{f, f'\})$.*

Interestingly, operator Γ is formally defined for the universe of sets of items, so that these relevancy results on single features can be directly extended to sets of features. This provides a proper generalization:

Definition 2 (Relevancy of feature sets). *Set of features $X \subseteq F$ is more relevant than set $Y \subseteq F$ iff $\Gamma^+(Y) = \Gamma^+(X \cup Y)$ and $\Gamma^-(X) = \Gamma^-(X \cup Y)$.*

To illustrate Definition 2 take the positive examples from Table 2, with negative data formed by classes none and hard together. Feature Spectacle=myope alone cannot be compared to feature Astigmatism=no alone with Definition 1 (because Astigmatism=no does not always imply Spectacle=myope in the negatives). For the same reason, Spectacle=myope cannot be compared to feature

Tear=normal alone. However, when considering these two features together, then Spectacle=myope turns out to be irrelevant w.r.t. the set {Astigmatism=no, Tear=normal}. So, the new semantic notion of Definition 2 allows us to decide if a set of features is structurally more important than another for discriminating the target class. In the language of rules: rule $Y \rightarrow +$ is *irrelevant* if there exists another rule $X \rightarrow +$ with $\Gamma^+(Y) = \Gamma^+(X \cup Y)$ and $\Gamma^-(X) = \Gamma^-(X \cup Y)$.

Moreover, from the structural properties of operator Γ and from Proposition 1, we can deduce that the semantics of relevant sets in Definition 2 is consistent:

Lemma 3. *A set of features $X \subseteq F$ is more relevant than set $Y \subseteq F$ (Definition 2) iff* $\mathrm{TP}(Y) \subseteq \mathrm{TP}(X)$ *and* $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$.

The forward proof of this result is based on Proposition 1, which ensures that $\mathrm{TP}(X \cup Y) \subseteq \mathrm{TP}(Y)$ when X is more relevant than Y. Moreover, since we have that $\Gamma^+(Y) = \Gamma^+(X \cup Y)$ by hypothesis, we can derive after a couple of steps that $\mathrm{TP}(Y) \subseteq \mathrm{TP}(X)$. Similarly, we can conclude $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$ for the negative part. The backward direction of Lemma 3 is also simple: if X and Y are two sets with $\mathrm{TP}(Y) \subseteq \mathrm{TP}(X)$ and $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$, we can imply after some deduction steps that $\mathrm{supp}^+(Y) = \mathrm{supp}^+(X \cup Y)$ and $\mathrm{supp}^-(X) = \mathrm{supp}^-(X \cup Y)$. By construction of Γ this means $\Gamma^+(Y) = \Gamma^+(X \cup Y)$ and $\Gamma^-(X) = \Gamma^-(X \cup Y)$.

3.1 Closed Sets for Discrimination

Together with the result of Lemma 3, it can be shown that only closed itemsets mined in the set of positive examples suffice for discrimination.

Theorem 1. *Let $Y \subseteq F$ be a set of features such that $\Gamma^+(Y) = X$ and $Y \neq X$. Then, set Y is less relevant than X (as in Definition 2).*[1]

The proof of this theorem is mainly based on the construction of Γ: $\Gamma^+(Y) = X$ ensures that $|\mathrm{TP}(Y)| = |\mathrm{TP}(X)|$; but because $Y \subseteq X$ it must be true that $\mathrm{TP}(Y) = \mathrm{TP}(X)$. This, together with Proposition 1 leads to X being more relevant than Y according to Definition 2.

Typically, in approaches such as Apriori-C [8], Apriori-SD [9] and RLSD [15] frequent itemsets with very small minimal support constraint are initially mined and subsequently post-processed in order to find the most suitable rules for discrimination. The new result presented here states that not all frequent itemsets are necessary: as shown in Theorem 1 only the closed sets have the potential to be relevant.

[1] We are aware that some generators Y of a closed set X might be exactly equivalent to X in terms of TP and FP, thus forming equivalence classes of rules. The result of this theorem characterizes closed sets in the positives as those representatives of relevant rules; so, any set which is not closed can be discarded, and thus, efficient closed mining algorithms can be employed for discrimination purposes. The next section will approach the notion of the shortest representation of a relevant rule, which will be conveyed by these mentioned equivalent generators.

To illustrate this result we use again the data in Table 2. There, we have $\Gamma^+(\{\text{Astigmatism=no}\} = \{\text{Astigmatism=no,Tear=normal}\}$. Thus, rule Astigmatism=no $\rightarrow$ + can be discarded: it covers exactly the same positives as {Astigmatism=no, Tear=normal}, but more negatives. Thus, a rule whose antecedent is {Astigmatism=no, Tear=normal} would be preferred for discriminating the class soft.

However, Theorem 1 simply states that closed itemsets suffice but some of them might not be necessary to discriminate the target class. It might well be that a closed itemset is irrelevant with respect to another closed itemset in the system. The next section is dedicated to the task of reducing the closure system of itemsets to characterize the final space of relevant sets of features.

4 Characterizing the Space of Relevant Sets of Features

This section studies how the dual closure system on the negative examples is used to reduce the lattice of closed sets on the positives. This reduction of the lattice will characterize a complete space of relevant sets of features for discriminating the target class. First of all, we raise the following two important remarks following from Proposition 1.

Remark 1. Given two different closed sets on the positives X and X' such that $X \not\subseteq X'$ and $X' \not\subseteq X$ (i.e., there is no ascending/descending path between them in the lattice), then they cannot be compared in terms of relevancy, since they cover different positive examples.

We exemplify Remark 1 with the lattice of Figure 1. The following two closed sets: {Age=young, Astigmatism=no, Tear=normal} and {Spectacle=myope, Astigmatism=no, Tear=normal}, are not comparable with subset relation: they cover different positive examples and they cannot be compared in terms of relevance.

Remark 2. Given two closed sets on the positives X and X' with $X \subset X'$, we have by construction that $\text{TP}(X') \subset \text{TP}(X)$ and $\text{FP}(X') \subseteq \text{FP}(X)$ (from Proposition 1). Notice that because X and X' are different closed sets in the positives, $\text{TP}(X')$ is necessarily a *proper* subset of $\text{TP}(X)$; however, regarding the coverage of false positives, this inclusion is not necessarily proper.

Remark 2 points out that two different closed sets in the positives, yet being one included in the other, may end up covering exactly the same set of false positives. In this case, we would like to discard the closed set covering less true positives. Because of the monotonicity property of support, the smaller one will be the most relevant. From these two remarks we have:

Theorem 2. *Let $X \subseteq F$ and $X' \subseteq F$ be two different closed sets in the positives such that $X \subset X'$. Then, we have that X' is less relevant than X (as in Definition 2) iff $\Gamma^-(X) = \Gamma^-(X')$.*

Table 3. The three closed sets corresponding to the space of relevant sets of features for data in Table 2

Occurrence list	Closed Set
1, 2, 3, 4, 5	{Astigmatism=no, Tear=normal }
2, 4, 5	{Spectacle=hypermetrope, Astigmatism=no, Tear=normal }
3, 4	{Age=pre-presbyopic, Astigmatism=no, Tear=normal }
1, 2	{Age=young, Astigmatism=no, Tear=normal }

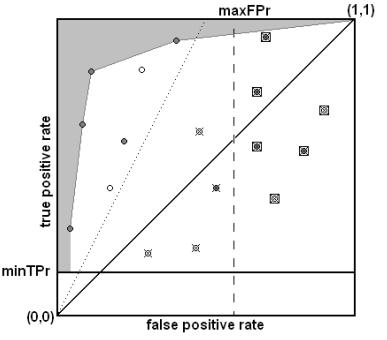


Fig. 2. The evaluation of relevant combinations of features in the ROC space

The forward direction of this proof is simple: when $X \subset X'$ then $X' = X' \cup X$, so that $\Gamma^+(X') = \Gamma^+(X' \cup X)$ and $\Gamma^-(X) = \Gamma^-(X' \cup X)$ gets trivial, thus satisfying Definition 2. The backward direction of the proof is also based on rewriting $\Gamma^-(X) = \Gamma^-(X')$ as $\Gamma^-(X) = \Gamma^-(X' \cup X)$, and $\Gamma^+(X')$ as $\Gamma^+(X') = \Gamma^+(X' \cup X)$; therefore, we also satisfy conditions of the definition.

Thus, by Theorem 2 we can reduce the closure system constructed on the positives by discarding irrelevant nodes: if two closed itemsets are connected by an ascending/descending path on the lattice of positives (i.e., they are comparable by set inclusion $\subset$), yet they have the same closure on the negatives (i.e., they cover the same false positives, or equivalently, their support on the negatives is exactly the same), then just the shortest set survives as a relevant set.

Finally, after Theorem 1 and Theorem 2, we can characterize the space of relevant sets of features for discriminating the selected target class as follows. These final sets can be directly interpreted as antecedents of discriminating rules.

Definition 3 (Space of relevant sets of features). *The space of relevant combinations of features for discriminating the target class is defined as those sets such that:* $\Gamma^+(X) = X$ *and there is no other closed set* $\Gamma(X') = X'$ *such that* $\Gamma^-(X') = \Gamma^-(X)$.

It is trivial to see after Remarks 1 and 2, that by construction, any two sets in this space cover always a different set of positives and a different set of negatives.

The three closed sets forming the space of relevant sets of features for the class soft are shown in Table 4. It can be checked that CN2 algorithm [3] would output the rule whose antecedent corresponds to the closed set in the first entry of Table 4; Ripper [4], would obtain the most specific relevant rules, i.e. those corresponding to the three last entries from Table 4. Finally, other algorithms such as Apriori-C would also output rules whose antecedents are not relevant such e.g. Astigmatism=no → Lenses=soft.

4.1 Shortest Representation of a Relevant Set

Based on Theorem 1 we know that generators Y of a closed set X are characterized to cover exactly the same positive examples, and at least the same negative examples. Because of this, any generator will be redundant w.r.t. its closure. However, we have $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$ for Y generator of X; so, it might happen that some generators Y are equivalent to their closed set X in that they cover exactly the same true positives and also the same false positives.

Definition 4. *Let $\Gamma^+(Y) = X$ and $Y \neq X$. We say that a generator Y is equivalent to its closure X if* $\mathrm{FP}(X) = \mathrm{FP}(Y)$.

The equivalence between true positives of Y and X is guaranteed because $\Gamma^+(Y) = X$. Therefore, it would be only necessary to check if generators cover the same false positives than its closure to check equivalence. Generators will provide a more general representation of the relevant set (because $Y \subset X$ by construction). So, $Y \rightarrow +$ is shorter than the rule $X \rightarrow +$ and it is up to the user to choose the more meaningful to her or to the application.

In terms of the closure operator of negatives, we have that Y is an equivalent generator of X iff $\Gamma^-(X) = \Gamma^-(Y)$.

5 Evaluation of Relevant Sets in the ROC Space

The ROC space is a 2-dimensional space that shows classifier (rule/ruleset) performance in terms of its *false positive rate* (also called 'false alarm'), $FPr = \frac{FP}{TN+FP} = \frac{FP}{|N|}$ plotted on the X-axis, and *true positive rate* (also called 'sensitivity') $TPr = \frac{TP}{TP+FN} = \frac{TP}{|P|}$ plotted on the Y-axis. The ROC space is appropriate for measuring the quality of rules, since rules with the best covering properties are placed in the top left corner, while rules that have similar distribution of covered positives and negatives as the distribution in the entire data set are close to the main diagonal.

The combinations of features of Definition 2 can be interpreted as condition parts of rules. Since they are induced with a minimal support constraint on the positives, they all lie above the minimum true positive rate constraint line (in Figure 2 denoted as minTPr). The rules removed by the relevancy filter are never those on the ROC convex hull (the empty circles are removed while the other remain). Furthermore, it can be trivially proved that we discover all the rules in the dataset on the ROC convex hull above the minimum true positives constraint (the full circles connected with a line). Therefore there are no rules outside the convex hull (grey area on the Figure 2 denotes an area without rules).

Sometimes an extra filtering criterion is required. In such cases we can imply a maximum FPr constraint covered by our relevant sets (in Figure 2 this constraint is represented by the dashed line, the rules eliminated by this constraint are shown in squares), or we can imply a minimum confidence constraint (represented by the dotted line, the rules eliminated by this constraint are crossed in Figure 2), or simply output rules on the convex hull, among others.

6 Experimental Evaluation

The presented theoretical study can be briefly summarized in the following steps:

- First, mining the set $S = \{X_1, \ldots, X_n\}$ of frequent closed itemsets from the target class (Theorem 1). This requires a minimum support constraint on true positives. Here we will use the efficient LCM algorithm [6].
- Second, reducing S to the space of relevant set of features by checking the coverage in the negatives (Theorem 2). Schematically, for any closed set $X_i \in S$, if there exists another closed set $X_j \in S$ s.t. both have same support in the negatives and $X_j \subset X_i$, then X_i is removed.

Finally, depending on the purpose of the application we can apply an extra filtering criterion, or compute minimal equivalent generators of the relevant sets as described above. For short, we will name this computing process as *RelSets* (for the Relevant Sets of features of Definition 2 we are discovering).

6.1 Emerging Patterns and Essential Rules on UCI Data

Emerging Patterns (EP) [5] are sets of features in the data whose supports change significantly from one class to another. More specifically, EPs are itemsets whose growth rates (the ratio of support from one class to the other, i.e. $\frac{TPr}{FPr}$ of the pattern) are larger than a user-specified threshold. In this experimental setting we want to show that some of the EPs mined by these approaches are redundant, and that our relevant sets correspond to the notion of compacted data representation for labeled data. Indeed, EPs are a superset of the result returned by RelSets.

In our comparisons we calculate relevant sets over a certain rate growth threshold (1.5 and infinite), and we compare this with the number of EPs by using the same rate growth constraint. Numerical attributes in the datasets are discretized when necessary by using four equal frequency intervals. Results are shown in the first part of Table 4.

Essential rules were proposed in [1] to reduce the number of association rules to those with nonredundant properties for classification purposes. Technically, they correspond to mining all frequent itemsets and removing those sets X s.t. there exists another frequent Y with $Y \subset X$ and having both the same support in positives and negatives. This differs from our proposal in the way of treating the positive class with closed sets. The compression factor we do for these rules is shown in the second part of Table 4. Note that essential rules are not pruned by rate growth threshold, and this is why their number is usually higher than the number of Emerging Patterns.

6.2 Subgroup Discovery in New Application Domains

Subgroup discovery [11,7] is a supervised descriptive induction task. The result of subgroup discovery is a set of subgroup descriptions (a rule set) that preferably has a low number of rules while each rule has high coverage and accuracy.

Table 4. Compression factor (CF% = $(1 - \frac{Relsets}{EPs}) \times 100$) of EPs and essential rules in UCI datasets. Note that we did not impose any minimum true positive threshold on any dataset, except for Lymphography and Crx, where all EPs, Relsets and essential rules were discovered with a 10% threshold on true positives. Also, note that in the second part of the table, essential rules and RelSets are not pruned by any rate growth threshold.

			EMERGING PATTERNS						ESSENTIAL RULES		
			Rate growth > 1.5			Rate growth ∞					
Dataset	Class	Distrib. %	EPs	RelSets	CF%	EPs	RelSets	CF%	Essence	RelSets	CF%
Lenses	soft	20.8	31	4	87.10	8	3	62.5	43	4	90.69
	hard	16.9	34	3	91.18	6	2	66.67	39	3	92.30
	none	62.5	50	12	76.00	42	4	9.52	89	19	78.65
Iris	setosa	33.3	83	16	80.72	71	7	90.14	76	20	73.68
	versicolor	33.3	134	40	70.15	63	10	84.13	111	41	63.06
	virginica	33.3	92	16	82.61	68	6	91.18	96	27	71.87
Breast-w	benign	65.5	6224	316	94.92	5764	141	97.55	3118	377	87.90
	malignant	34.5	3326	628	81.12	2813	356	87.34	2733	731	73.25
SAheart	0	34.3	4557	1897	58.37	2282	556	75.64	6358	4074	35.92
	1	65.7	9289	2824	69.60	3352	455	86.43	9622	4042	58
Balance-scale	B	7.8	271	75	72.32	49	49	0.00	415	147	88.67
	R	46	300	84	72.00	90	90	0.00	384	364	5.20
Yeast	MIT	16.4	3185	675	78.81	250	40	84.00	2258	1125	50.17
	CYT	31.2	3243	808	75.08	68	16	76.47	2399	1461	80.78
	ERL	0.3	1036	5	99.52	438	4	99.09	417	5	98.80
Monk-1	0	64.3	1131	828	26.79	321	18	94.39	1438	1135	21.07
	1	35.7	686	9	98.69	681	4	99.41	1477	363	75.42
Lymphography	metastases	54.72	36435	666	98.17	10970	90	99.18	1718	369	78.52
10% min supp.	malign	41.21	61130	740	98.79	19497	55	99.72	2407	476	80.22
Crx	+	44.5	3366	782	76.76	304	26	91.44	2345	1091	53.47
10% min supp.	−	55.5	3168	721	77.24	12	5	58.33	2336	1031	55.86

Table 5. Comparison of algorithms RelSets and SD on new subgroup discovery problems. Column RelSets-ROC shows the number of RelSets rules on the ROC convex hull.

Dataset	Class	Num. of rules			AUC		Time	
		RelSets	RelSets-ROC	SD	RelSets	SD	RelSets	SD
potato	sensitive	1	1	20	100%	100%	<1s	>1h
microarray	resistant	1	1	20	100%	91%	<1s	>1h
dribble	kick	110	7	20	89%	61%	<1s	3min
pass	pass	8	4	0	88%	0%	<1s	3min
	shoot	1	1	20	100%	100%	<1s	3min

The first experiment is performed on a real life potato microarray dataset with high dimensionality on the number of attributes. The goal is to distinguish between two different classes of resistance of four transgenic potato lines. After data preprocessing, we have only 12 examples (6 virus resistant and 6 virus sensitive examples) and 19,131 attributes. In Table 5 it can be seen how slowly the subgroup discovery algorithm SD performs and that RelSets performs better in terms of the area under the ROC curve (AUC) in the ROC space. Moreover, standard subgroup discovery algorithms present subgroups that are not as satisfactory to end users as subgroups found by RelSets.

The second experiment was performed on a real world strategy learning problem of robots playing soccer. In this dataset we have four classes: three classes represent successful moves made by the robots and the majority class (92%) when nothing interesting happens. We ran RelSets with a minimum true positive rate constraint of 20%. In Table 5 we show that we do not only outperform the algorithm SD in time, but also in quality (area under ROC convex hull).

7 Conclusions

We have presented a theoretical framework that, based on the covering properties of closed itemsets, characterizes those sets of features that are relevant for discrimination. We call them closed sets for labeled data, since they keep similar structural properties of classical closed sets, yet taking into account the positive and negative dimension of examples. In practice the approach shows major advantages for: compacting Emerging Patterns and essential rules and solving hard subgroup discovery problems. Thresholds on positives make the method tractable even for large databases with many features. Future work may adapt efficient algorithms of EPs in [5] for discovering relevant sets.

References

1. E. Baralis and S. Chiusano. Essential classification rule sets. *ACM Trans. Database Syst.*, 29(4):635–674, 2004.
2. Stephen D. Bay and Michael J. Pazzani. Detecting group differences: Mining contrast sets. *Data Min. Knowl. Discov.*, 5(3):213–246, 2001.
3. P. Clark and T. Niblett. The CN2 induction algorithm. *Mach. Learn.*, 3(4):261–283, 1989.
4. W.W. Cohen. Fast effective rule induction. In *Proc. 12th Int. Conf. on Machine Learning*, pages 115–123, 1995.
5. G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. In *KDD '99: Proc. of the fifth ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 43–52, 1999.
6. B. Goethals and M. Zaki. Advances in frequent itemset mining implementations: report on fimi'03. *SIGKDD Explor. Newsl.*, 6(1):109–117, 2004.
7. D. Gramberger and N. Lavrač. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.
8. V. Jovanoski and N. Lavrač. Classification rule learning with APRIORI-C. In *EPIA '01*, pages 44–51. Springer-Verlag, 2001.
9. B. Kavšek and N. Lavrač. APRIORI-SD: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, To appear, 2006.
10. N. Lavrač, D. Gamberger, and V. Jovanoski. A study of relevance for learning in deductive databases. *Journal of Logic Programming*, 40(2/3):215–249, 1999.
11. N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.
12. N. Pasquier, Y. Bastide, R. Taouil L., and Lakhal. Closed set based discovery of small covers for association rules. In *Proc. ICAD*, pages 361–381, 1999.
13. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java implementations.* Morgan Kaufmann, 2005.
14. M. Zaki. Mining non-redundant association rules. *Data Mining and Knowledge Discovery: An Int. Journal*, 4(3):223–248, 2004.
15. J. Zhang, E. Bloedorn, L. Rosen, and D. Venese. Learning rules from highly unbalanced data sets. In *ICDM'04*, pages 571–574, 2004.

Finding Trees from Unordered 0–1 Data

Hannes Heikinheimo, Heikki Mannila, and Jouni K. Seppänen

HIIT Basic Research Unit, Lab. Computer and Information Science,
FI-02015 Helsinki University of Technology, Finland
{Hannes.Heikinheimo, Heikki.Mannila, Jouni.Seppanen}@tkk.fi

Abstract. Tree structures are a natural way of describing occurrence relationships between attributes in a dataset. We define a new class of tree patterns for unordered 0–1 data and consider the problem of discovering frequently occurring members of this pattern class. Intuitively, a tree T occurs in a row u of the data, if the attributes of T that occur in u form a subtree of T containing the root. We show that this definition has advantageous properties: only shallow trees have a significant probability of occurring in random data, and the definition allows a simple levelwise algorithm for mining all frequently occurring trees. We demonstrate with empirical results that the method is feasible and that it discovers interesting trees in real data.

1 Introduction

Frequent pattern discovery has been extensively studied, especially in the case of 0–1 data, where various algorithms exist for mining frequent itemsets and association rules. Here we propose a new class of co-occurrence patterns: trees. The idea is to search for hierarchies of general and more specific attributes. For example, consider document data and a tree with attribute A as the root and B and C as the children of A, and D as the child of B (see Figure 1). This means that A is the general concept, B and C are more specific terms related to A, and D is a further specialization of B.

An observed row u follows the hierarchy described by the tree if the attributes in the tree that are 1 in u form a subtree of T containing the root. In the example

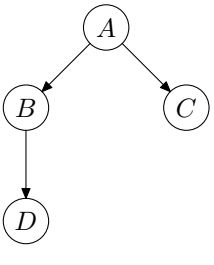


Fig. 1. Example tree

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 175–186, 2006.
© Springer-Verlag Berlin Heidelberg 2006

tree of Figure 1 a row with $u(A) = u(B) = u(C) = 1$ and $u(D) = 0$ satisfies this condition, but a row with $u(A) = u(D) = u(C) = 1$ and $u(B) = 0$ does not.

We consider the task of finding trees T such that there are sufficiently few rows violating the subtree condition (few *conflicts*). To prevent trivial trees consisting only of attributes occurring very rarely, we additionally require that each attribute occurring in T has a high enough frequency in the data. The task we consider is, given two thresholds τ and σ, to find all trees that have at most τ conflicts and each attribute occurring in the tree has frequency at least σ.

While there has been lots of research on finding trees from tree- or graph-structured data (see, e.g., [1,2,3,4,5,6]), the crucial difference is that we start from unstructured 0-1 data, as in mining frequent sets and association rules. Hierarchical clustering or finding phylogenetic trees (see, e.g., [7]) looks for finding trees from 0-1 data, but typically the goal is to find one tree containing all the attributes of the data. The same is true for finding tree-structured Bayes nets from data. Another difference is that in a hierarchical clustering or phylogenetic tree all attributes of the data are in the leaves of the tree. We seek trees where all nodes of the tree are items from the data.

Another somewhat analogous class of patterns are approximate itemsets, such as error-tolerant or dense itemsets [8,9,10]. These are relaxed versions of frequent itemsets: the set is considered to occur in a row provided most of the attributes of the set are equal to 1 in the row. Similarly to these patterns, in this work a tree can be supported by rows that do not have all of the items of the tree. The tree structure reflects more closely the kinds of co-occurrence that are in fact present in the data. Fragments of order [11] are a type of directed itemsets: a fragment is violated by rows having two of its items but lacking at least one item that appear between the two. Fragments of order can be viewed as simple unrooted trees having only one branch.

The rest of this paper is structured as follows. In Section 2 we present the definition of tree patterns. We discuss their theoretical properties in Section 3. In Section 4 we give algorithms for discovering trees, and present two measures for selecting interesting trees. Empirical results demonstrating that the method is feasible and that it finds interesting trees are given in Section 5. Extensions to the pattern class are discussed in Section 6, and Section 7 is a short conclusion.

2 Tree Patterns and 0–1 Data

Let R be a set of 0-1 valued attributes (also called items). A 0-1 dataset D over R is a table of rows of 0s and 1s with R as the set of column headers. The dataset D can be also considered as an unordered multiset of rows u, where each row is a subset of R. We denote attributes by the letters A, B, and C, and by $u(A)$ the value 1 or 0, according to whether the attribute A is present in u or not. We denote by n the number of rows in D and by m the number of attributes, i.e., $|R|$. The *frequency* $f(A)$ of an attribute A is the fraction of rows u such that $u(A) = 1$.

A *tree* is a pair $T = (A, \mathcal{C})$, where A is an attribute and $\mathcal{C} = \{ T_1, T_2, \ldots, T_k \}$ is the set of subtrees, each of which is a tree. (For leaves of the tree the set $\mathcal{C}$

is empty.) We say that A is the root of the tree. Each T_i has the form $T_i = (A_i, \mathcal{C}_i)$; the attributes $A_1, \ldots, A_k$ are called the children of A (this phrasing is unambiguous as we require that each attribute appears in the tree at most once). An attribute B is a descendant of A (or, equivalently, A is an ancestor of B) either if B is a child of A or B is a descendant of a child of A. The set of nodes of the tree T are simply all attributes occurring in the tree.

The *conflict count* $c(T, D)$ of T in D is the number of rows $u \in D$ such that there is a node A and its descendant B such that $u(A) = 0$ and $u(B) = 1$. Given thresholds τ and σ, both in $[0, 1]$, the collection of trees $\mathcal{TP}(D, \tau, \sigma)$ consists of all trees T such that $c(T, D) \leq \tau n$ and $f(A) \geq \sigma n$ for all A occurring in T, where n is the number of rows in the dataset D.

The computational problem we consider in this paper is the following.

Problem 1. Given D, σ, and τ, compute $\mathcal{TP}(D, \tau, \sigma)$.

Note that in the definition of $\mathcal{TP}(D, \tau, \sigma)$ we use an *upper* bound τ on the number of conflicts, while frequent patterns typically are defined by a *lower* bound on the number of occurrences. Our parameter σ has this role, preventing attributes with very low frequency from being considered.

3 Basic Properties of Tree Patterns

In this section we consider the basic properties of tree patterns and the pattern collection $\mathcal{TP}(D, \tau, \sigma)$.

Monotonicity. The first observation is simple monotonicity property typical for frequent patterns. A tree S is a rooted subtree of tree T, if S can be obtained from T by a series of removals of leaves. The following proposition is immediate.

Proposition 1. *The pattern class* $\mathcal{TP}(D,\tau,\sigma)$ *is monotone with respect to rooted subtrees, i.e., if* $T \in \mathcal{TP}(D, \tau, \sigma)$ *and* S *is a subtree of* T*, then* $S \in \mathcal{TP}(D, \tau, \sigma)$.

Trees and Association Rules. Next we discuss the relationship of trees and association rules. For a simple tree T containing root A and a child B, the conflict count $c(T)$ is $n(f(B) - f(AB))$, where $f(AB)$ is the frequency of the attribute set AB, i.e., the relative number of rows u with $u(A) = u(B) = 1$. Noting that $\gamma = f(AB)/f(B)$ is the accuracy (confidence) of the association rule $B \to A$, we have that $c(T) = nf(B)(1 - \gamma)$. One can ask whether other, more complex trees could be reduced to association rules? Could we perhaps find all the trees in $\mathcal{TP}(D, \tau, \sigma)$ just by postprocessing a set of association rules between attributes? (See, e.g., [12,13,14,15,16] for interesting work on postprocessing collections of association rules.) This turns out not to be the case, however.

A row u satisfies the subtree condition for tree T if the rule $C \to D$ is true on u for all pairs (C, D) such that C is a descendant of D in T. However, there is no simple formula for the conflict count of a tree T given the accuracies of the rules $C \to D$ for attributes occurring in T. The reason is that a tree conflicts with a row u if at least one rule is violated: the frequency under which this happens

depends on the interaction of the different rules. As an example, consider the tree with root A, and B and C as the children of A. If $f(A) = f(B) = f(C) = 0.2$ and $f(AB) = f(AC) = 0.1$, the accuracies of the rules $B \to A$ and $C \to A$ are both 0.5. The conflict count of the tree T can, however, vary between $0.2n0.5$ and $2(0.2n0.5)$, depending on whether the conflicts are on the same rows or not. Hence there is no algorithm for computing $\mathcal{TP}(D, \tau, \sigma)$ that would take as input only the association rules between attributes.[1]

Number of Possible Trees. The number of rooted labeled trees on m vertices is m^{m-1}. This follows from a theorem of Cayley, which states that the number of labeled trees is m^{m-2}; see e.g. [17, Section 3.3] or [18, sequence A000169]. The number of possible roots is of course m, from which the result follows. This implies that it would be infeasible to consider all trees even for moderate values of m.

Trees in Random Data. We next address the question how many trees are in the collection $\mathcal{TP}(D, \tau, \sigma)$ in random data, similarly to the discussion in [19] on frequent itemsets. Suppose D contains independent and identically distributed entries, with probability p of a 1 for each entry. Trees that only have a root and k children cause a conflict only if the value of the attribute of the root is 0; thus the conflict probability is fairly low.

For trees with longer branches it is straightforward to demonstrate that the probability q of a conflict grows fast. Using Chernoff bounds it is then easy to show that the probability that a tree with longer branches has less than, say, $nq/2$ conflicts is at most $\exp(-cnq)$ for some constant c. There are $P = \binom{m}{k} k^{k-1}$ trees with k nodes selected from the set of m attributes, implying that the expected number of trees with less than $nq/2$ conflicts is thus bounded by $P \exp(-cnq)$. We have $\log P \leq k(\log m + \log k) \leq 2k \log m$. Thus, if $2k \log m \leq cnq$, the expected number of trees (with sufficiently long branches) in $\mathcal{TP}(D, \tau, \sigma)$ is at most 1. We omit the details.

4 Generating the Collection $\mathcal{TP}(D, \tau, \sigma)$

4.1 The Levelwise Algorithm

Proposition 1 allows using a standard levelwise algorithm for computing trees in $\mathcal{TP}(D, \tau, \sigma)$, as when computing frequent itemsets: start from single attributes, and on every pass combine trees of size k into trees of size $k + 1$. However, the combination phase is not as simple as for itemsets.

One approach would be to try adding every attribute into every possible position in each tree, but with a large number of attributes this would force the algorithm to consider prohibitively many candidate trees. Another possibility is to try combining all pairs of trees, which takes quadratic time in the number

[1] The exponential collection of frequencies of *all* frequent sets for frequency threshold 0 specify the distribution of the data rows uniquely, so that exponential input would suffice to determine also the collection $\mathcal{TP}(D, \tau, \sigma)$.

of trees. Instead, we use the approach of Zaki [6]. Briefly, a tree is represented as a string by traversing it depth-first in preorder, recording the attribute in each node, and -1 when backtracking. For example, the tree in Figure 1 would be encoded as $(A, B, D, -1, -1, C, -1)$. In this encoding, it is sufficient to consider combining pairs of trees sharing the same $(k-1)$-prefix, which limits the quadratic behavior to much smaller sets than the complete set of k-trees. For details, see [6]; note that since we restrict attributes to occur at most once in each tree, not all combinations listed by Zaki are needed.

A drawback of the combination method is that each $(k+1)$-tree is generated multiple times as isomorphic copies: for example, the isomorphic trees $(A, B, -1, C, -1)$ and $(A, C, -1, B, -1)$ get both generated. We optimize the database pass by only accessing the database for trees where the children of each node are in alphabetical order, and using the same information for isomorphic trees. However, it is not possible to completely prune the copies. For example, the 4-tree $(A, C, -1, D, B, -1, -1)$ can only be generated from the 3-trees $(A, C, -1, D, -1)$ and $(A, C, -1, B, -1)$, the latter of which does not have the order property. Another possibility would be to work only with some canonical forms of trees, as in [4].

The size of the class $\mathcal{TP}(D, \tau, \sigma)$ is highly sensitive to the values of the two parameters. A way to ameliorate this problem is to use a top-k algorithm similar to that developed for dense itemsets [10].

We remarked in the previous section that in random data trees of small depth can have low conflict count. It is straightforward to modify the above algorithm to construct only, e.g., binary trees, thus guaranteeing longer branches. We omit the details.

4.2 Selecting Interesting Trees

The collection $\mathcal{TP}(D, \tau, \sigma)$ will in many cases be quite large, and tools are needed for selecting the most interesting trees.

We present two measures for selecting interesting trees: specificity and conflict ratio. The *specificity* $\phi(T)$ of a tree T is the size of the transitive closure of the tree, when it is viewed as a relation on the set of attributes. In other words, it is the number of (ancestor, descendant) pairs in the tree. A single-branch tree has maximal specificity and a shallow tree whose leaves are the children of the root has minimal specificity. The *conflict ratio* $E[c(T)]/c(T)$ of a tree T is obtained by comparing the number $c(T)$ of conflicting rows in the data to its expectation $E[c(T)]$ under the assumption that the attributes are independent and have the marginal frequencies observed in the data.

The expectation $E[c(T)]$ can be computed recursively for each tree. For $T = (A, \mathcal{C})$, the probability of no conflict, under the independence assumption, is

$$\begin{aligned} 1 - \Pr(u \text{ conflicts with } T) = {} & \Pr(u(A) = 0) \prod_{B} \Pr(u(B) = 0) \\ & + \Pr(u(A) = 1) \prod_{S \in \mathcal{C}} (1 - \Pr(u \text{ conflicts with } S)), \end{aligned}$$

where the first product is taken over all attributes B represented in T, except for the root A, and the second over all child-trees S of T.

The conflict ratio will be high for trees that have much fewer conflicts than would be expected under the independence assumption, i.e., trees that capture interesting co-occurrence patterns in the data. The two interestingness measures can be used to rank the trees in various ways.

5 Experiments

In this section we report on the experimental results we have obtained on generated and real data. Due to space constraints we discuss the results only briefly.

5.1 Generated Data

We generated data using the following procedure. First, a number of disjoint trees with different values of the specificity measure were created by hand. The number of trees used for the experiment was varied by taking different subcollections of the trees. Given such a collection $\mathcal{S}$, data was produced as follows. A row u was generated by first making all attributes of u equal to 0. Then, each tree $T \in \mathcal{S}$ was selected with probability p. If T was selected, we sampled a subset X of the nodes of T by taking each node with probability q. Letting Y be the set of all ancestors of nodes in X, we let $u(A) = 1$ for all $A \in X \cup Y$. Finally, each bit in the dataset was flipped independently with probability r to create noise. The parameter values used were $p = q = 0.5$ and $r = 0.1$, and 1000 data rows were generated.

From the generated data, trees were mined using a Java implementation of the levelwise algorithm described in Section 4. The parameter σ was chosen to be 0.2, since each tree has a $p = 0.5$ chance of occurring, and each attribute in the tree may have as low as a $pq = 0.25$ chance of occurring. The parameter τ was chosen as large as possible so that a reasonable number of trees was still obtained; typically $\tau = 0.3$ was close to the limit. Selecting the parameters was also the reason that only disjoint trees were considered. With overlapping trees large numbers of shallow subtrees are generated, and the conflict threshold has to be quite low. This prevents the discovery of the more interesting trees. With disjoint trees, the result set $\mathcal{TP}$ was reasonably small, while containing all the trees in $\mathcal{S}$. In order to test how well these trees were positioned in the results with respect to the two interestingness measures, the mined trees were partitioned into classes by their specificity ϕ, and each class was sorted by the conflict ratio $E[c]/c$. From each specificity class, trees were selected into the final result set $\mathcal{R}$ in decreasing order of conflict ratio until all trees in $\mathcal{S}$ had been selected. The size $|\mathcal{R}|$ of the final result set is thus a measure of how close to the top the generating trees in $\mathcal{S}$ were. The results are shown in Table 1.

We see that in every case the number of trees one needs to examine in order to find the generating trees is fairly low. In fact, most of the extra trees are variations of the generating trees: for example, in the smallest data set, one of

the two generating trees is found immediately, and three of its simple variants precede the other generating tree in conflict ratio order. The sensitivity to the parameter is evident in that while we have $\sigma = 0.2$ and $\tau = 0.3$ in all the cases shown the size of the output $|\mathcal{TP}|$ varies non-monotonically in m.

5.2 Real Data

We used three real datasets in our experiments: data about terms used in NFS abstracts [20], a database about students and courses at the Computer Science Department of the University of Helsinki, and paleontological data [21,22]. We present the results only for the first two data sets; the results were similar for the third one.

Abstracts Data. The data set [20] consists of 128820 abstracts describing NSF awards for basic research. The observations correspond to the abstracts and the variables correspond to the terms occurring in them. For preprocessing we applied the Porter Stemming Algorithm [23] to merge variables corresponding to terms with a common stem. In addition, we reduced the dataset by taking a random 2% sample of the observations and choosing 66 subjectively interesting stem terms[2] for the experiments. The final preprocessed data set consisted of 2510 observations (rows) and 66 variables (columns), with a total of 11262 entries of 1s and an average of 4.5 1s per row.

Table 2 shows the number of trees obtained for different conflict thresholds τ and frequency thresholds σ. We see that the number of elements in the answer increases rapidly with decreasing frequency threshold σ and increasing conflict threshold τ. One should observe, however, that it is quite easy to iteratively find values of σ and τ that produce outputs of desired size.

When inspected visually, the resulting trees look intuitive. As an example, we found the tree depicted in Figure 2(a), at a conflict count frequency of 13.1% and a conflict ratio of 1.58. With another set of parameters, we found the very intuitive tree in Figure 2(b) at a conflict count frequency of 21.2% and a conflict ratio of 1.19. The tree in Figure 2(a) has specificity 5, whereas the tree in Figure 2(b) has specificity 6.

Course Enrollment Data. The data consists of course enrollment records for courses held at the Department of Computer Science at the University of Helsinki. The data set has 3506 observations corresponding to students and 98 variables corresponding to courses. The mean number of 1s per row is 4.6. The

[2] The terms chosen were algebra, algorithm, atom, behavior, biolog, carbon, cell, cellular, channel, chemistri, climat, code, comput, distribut, dna, document, earth, ecolog, ecosystem, educ, electron, energi, environment, enzym, evolut, genet, geolog, hardwar, internet, isotop, life, light, link, magnet, materi, mathemat, matter, metal, molecular, morpholog, natur, network, nonlinear, nuclear, numer, ocean, oxid, physic, pi, plasma, protein, quantum, record, scienc, semiconductor, social, softwar, statist, surfac, temperatur, theoret, topolog, transit, transport, water and web.

Table 1. Results on generated data. m: number of attributes; $|\mathcal{S}|$: number of trees used in the generating process; σ, τ: thresholds for frequency and conflicts; $|\mathcal{TP}|$: size of the output set; $|\mathcal{R}|$: size of the final result set obtained by taking enough trees to cover $\mathcal{S}$.

m	$\lvert\mathcal{S}\rvert$	σ	τ	$\lvert\mathcal{TP}\rvert$	$\lvert\mathcal{R}\rvert$
10	2	0.2	0.30	1343	5
14	3	0.2	0.30	1172	8
18	4	0.2	0.30	1965	21
20	5	0.2	0.30	2208	27
23	6	0.2	0.30	1674	45
28	7	0.2	0.30	5469	43

Table 2. Results for the abstracts data set. The number of trees in the collection $\mathcal{TP}(D, \tau, \sigma)$ for various values of τ and σ. $k = \max |T|$, the largest tree; $h = \max \phi(T)$ the maximal specificity.

τ	σ	$\lvert\mathcal{TP}\rvert$	k	h	cand	time/sec.
0.14	0.12	26	3	2	194	0.573
0.16	0.12	33	3	3	250	0.603
0.18	0.12	51	3	3	397	0.835
0.14	0.08	649	3	3	12514	5.284
0.16	0.08	1894	3	3	39091	19.22
0.18	0.08	3890	4	4	108825	96.05
0.14	0.06	6637	5	6	204741	339.1
0.16	0.06	14927	5	7	538334	1970
0.18	0.06	48176	6	7	1683841	30300

total number of 1s in the data is 16086. Table 5.2 shows the number of trees obtained for different conflict and frequency thresholds τ and σ.

For the course enrollment data there is an ordering in which the department recommends the students to take some of the courses. For instance, some courses require only basic understanding of programming concepts, whereas some of the courses have more specific prerequisites. As an example of this, we found the tree depicted in Figure 3 at a conflict threshold of 16.3% and a high conflict ratio of 2.20. The tree reflects the fact that the more advanced courses Data structures and Programming in C have Java programming as a prerequisite, whereas for Computer Organization the course Introduction to Programming suffices. This is also the order in which the department recommends these courses to be taken.

Outlier Detection. To evaluate the usefulness of discovered trees we experimented with their use in outlier detection. An observation that conflicts with several of the strongest tree patterns is likely to be an outlier. As a test of this, we performed the following experiment. We took the course enrollment data set,

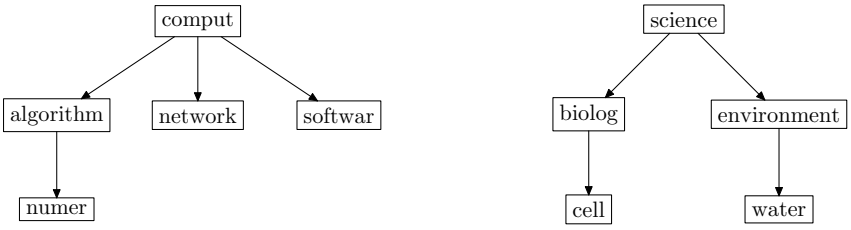


Fig. 2. Two example trees in abstracts data

Table 3. Results for the course enrollment data. The number of trees in the collection $\mathcal{TP}(D, \tau, \sigma)$ for various values of τ and σ. $k = \max |T|$, the largest tree; $h = \max \phi(T)$ the maximal specificity.

τ	σ	$\lvert\mathcal{TP}\rvert$	k	h	cand	time/sec.
0.14	0.12	68	3	3	581	0.524
0.16	0.12	124	4	3	991	0.795
0.18	0.12	220	4	5	1830	1.163
0.14	0.1	262	4	5	1929	1.318
0.16	0.1	484	5	6	4128	2.733
0.18	0.1	998	5	7	9367	6.373
0.14	0.08	3955	6	8	71982	145.6
0.16	0.08	11475	7	10	281398	3310
0.18	0.08	35398	8	12	1221143	93740

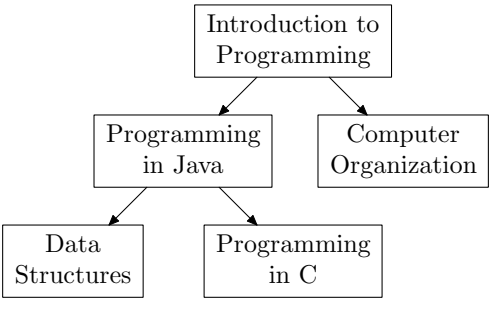


Fig. 3. Example tree found in course enrollment data

here denoted by D, and generated an additional data set N with independent attributes with the same marginal frequencies as in D. The size of N was 5% of the original data D. Then, frequent trees $\mathcal{TP} = \mathcal{TP}(E, \tau, \sigma)$ were mined from the augmented data $E = D \cup N$ with $\tau = 0.14$ and $\sigma = 0.08$. From each specificity class, the 30 trees with maximal conflict ratio were selected to form

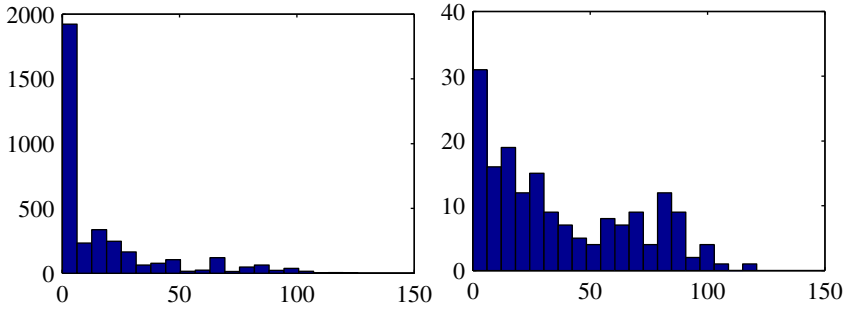


Fig. 4. Histogram for the conflict count between the rows in the data and the 164 generated trees. The figure on the left depicts the histogram for the real rows in the data (total 3506). The figure on the right depicts the histogram for the 5% added noise rows (total 175). The original data rows have fewer conflicts with the chosen 164 trees than the added rows. See the text for details.

a subset $\mathcal{S} \subset \mathcal{TP}$. The specificities ranged from 1 to 6, but since the generated number of trees with specificity 5 and 6 was 23 and 21 a total of 164 trees were selected to $\mathcal{S}$. For each row $u \in E$, we determined how many trees $T \in \mathcal{S}$ conflict with u.

The result was that for rows $u \in D$, the average conflict count was 16.8, with a standard deviation of 24.5, and for $u \in N$, the average was 37.8 with a standard deviation of 31.2. Figure 4 shows the histograms of the number of conflicts per row for the real data D and for the added rows N. Thus, the noise rows behave clearly differently from the real data rows from the viewpoint of tree patterns.

6 Extensions

A problem clearly seen in the experiments is that a large number of small, shallow trees crop up and slow the algorithm down before it has investigated more interesting trees. A possible solution is to restrict the class of trees considered. For example, if the number of children of each node is restricted to be at most two, a levelwise algorithm would still be possible, and there would be much fewer shallow trees to consider.

Another way to approach the problem is to change the search strategy from the levelwise, breadth-first search. In the case of frequent itemsets, there are depth-first algorithms for mining the maximal frequent itemsets without considering all their subsets (see, e.g, [24] and [25]). Adapting such algorithms for trees is an interesting direction for future research.

If more efficient search strategies are developed, it would also be interesting to broaden the class of patterns: directed acyclic graphs would be a natural generalization of trees.

7 Conclusion

We have introduced the idea of mining trees from unordered 0–1 data and shown that this pattern class is distinct from traditional frequent itemsets or association rules. We have shown empirically that the levelwise algorithm can find interesting trees in both generated and real data. In real data, our experiments show that there are interesting co-occurrence patterns that are naturally captured as trees.

References

1. Chi, Y., Muntz, R.R., Nijssen, S., Kok, J.N.: Frequent subtree mining – an overview. Fundamenta Informaticae **66** (2005) 161–198
2. Chi, Y., Yang, Y., Muntz, R.R.: Indexing and mining free trees. In: Proceedings of the Third IEEE International Conference on Data Mining (ICDM). (2003) 509 – 512
3. Chi, Y., Yang, Y., Muntz, R.R.: Mining frequent rooted trees and free trees using canonical forms. Technical Report CSD-TR No. 030043, UCLA Computer Science Department (2003) `ftp://ftp.cs.ucla.edu/tech-report/2003-reports/030043.pdf`.
4. Chi, Y., Yang, Y., Muntz, R.R.: HybridTreeMiner: an efficient algorithm for mining frequent rooted trees and free trees using canonical forms. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM). (2004) 11 – 20
5. Nijssen, S., Kok, J.N.: Efficient discovery of frequent unordered trees. In: First International Workshop on Mining Graphs, Trees and Sequneces (MGST). (2003) 55 – 64
6. Zaki, M.J.: Efficiently mining frequent trees in a forest. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2002) 71 – 80
7. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Inc., Sunderland, MA (2004)
8. Pei, J., Tung, A.K., Han, J.: Fault-tolerant frequent pattern mining: Problems and challenges. In: Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD). (2001) 7–12
9. Yang, C., Fayyad, U., Bradley, P.S.: Efficient discovery of error-tolerant frequent itemsets in high dimensions. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2001) 194 – 203
10. Seppänen, J.K., Mannila, H.: Dense itemsets. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2004) 683–688
11. Gionis, A., Kujala, T., Mannila, H.: Fragments of order. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2003) 129–136
12. Tuzhilin, A., Adomavicius, G.: Handling very large numbers of association rules in the analysis of microarray data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2002) 396–404
13. Lent, B., Swami, A.N., Widom, J.: Clustering association rules. In: Proceedings of the 13th International Conference on Data Engineering (ICDE). (1997) 220–231

14. Liu, B., Hsu, W., Ma, Y.: Pruning and summarizing the discovered associations. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (1999) 125–134
15. Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A.I.: Finding interesting rules from large sets of discovered association rules. In: Proceedings of the Third International Conference on Information and Knowledge Management (CIKM). (1994) 401–407
16. Jaroszewicz, S., Simovici, D.A.: Pruning redundant association rules using maximum entropy principle. In: Proceedings of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD). (2002) 135–147
17. Kreher, D.L., Stinson, D.R.: Combinatorial Algorithms: Generation, Enumeration and Search. Discrete mathematics and its applications. CRC Press (1999)
18. Sloane, N.J.A.: The on-line encyclopedia of integer sequences (2006) `http://www.research.att.com/ njas/sequences/`.
19. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in Knowledge Discovery and Data Mining. AAAI Press (1996) 307–328
20. Hettich, S., Bay, S.D.: The UCI KDD archive (1999) Irvine, CA: University of California, Department of Information and Computer Science. `http://kdd.ics.uci.edu`.
21. Fortelius, M. (coordinator): Neogene of the old world database of fossil mammals (NOW) (2006) University of Helsinki. `http://www.helsinki.fi/science/now/`.
22. Fortelius, M., Gionis, A., Jernvall, J., Mannila, H.: Spectral ordering and biochronology of european fossil mammals. Paleobiology **32** (2006) 206–214
23. Porter, M.F.: An algorithm for suffix stripping. Program **14** (1980) 130–137
24. Bayardo, R.: Efficiently mining long patterns from databases. In: Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD). (1998) 85–93
25. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000)

Web Communities Identification from Random Walks

Jiayuan Huang[1,2], Tingshao Zhu[2], and Dale Schuurmans[2]

[1] University of Waterloo, Waterloo, Canada
[2] University of Alberta, Edmonton, Canada

Abstract. We propose a technique for identifying latent Web communities based solely on the hyperlink structure of the WWW, via random walks. Although the topology of the Directed Web Graph encodes important information about the content of individual Web pages, it also reveals useful meta-level information about user communities. Random walk models are capable of propagating local link information throughout the Web Graph, which can be used to reveal hidden global relationships between different regions of the graph. Variations of these random walk models are shown to be effective at identifying latent Web communities and revealing link topology. To efficiently extract these communities from the stationary distribution defined by a random walk, we exploit a computationally efficient form of directed spectral clustering. The performance of our approach is evaluated in real Web applications, where the method is shown to effectively identify latent Web communities based on link topology only.

1 Introduction

Increasingly, the World Wide Web is playing an important role in peoples' lives as a main destination for information. However, the sheer heterogeneity of Web users and authors—given diverse backgrounds and interests—hampers traditional information retrieval approaches that rely on content analysis alone. The Web is comprised of multiple communities [5] created by different groups of people having common interests. The identification of Web communities can help users with their information retrieval goals, by allowing the construction of preclassified directories and the creation of more effective recommendation services. Random walk models have been successfully used for Web ranking in the past [12,11], and have also raised interest in identifying Web communities.

In this paper we investigate how the *directed* hyperlink information conveyed via random walks can help one efficiently identify latent Web communities from the hyperlink topology alone. Our work on identifying Web communities exploits recent progress on *directed* spectral clustering [16], and contributes further understanding to the nature of such clustering techniques. Here, we analyze directed spectral clustering from a random walk perspective.

Intuitively, a coherent Web community can be identified by a subset of Web pages that is strongly connected within the subset, while only being weakly connected with pages outside the subset. We assume that if two pages are directly

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 187–198, 2006.
© Springer-Verlag Berlin Heidelberg 2006

linked, their interests are assumed to be somewhat related [7]. We also take *co-citation* [15] and *co-reference* [9] relations into account to accurately identify latent Web communities. Such high level connections provide useful relationship information, since sometimes connections between Web pages might not be as obvious as such direct links.

For Web community identification, we first examine a one-step random walk model that captures low level aspects of hyperlink connectivity. We then consider a two-step random walk model with different variations that captures higher level information by exploiting the existence of co-reference and co-citation relationships in the link topology. For the two-step model, we introduce a damping process that samples the entire Web uniformly, which allows the random walk to be properly applied to general Web graphs.The selection of the damping factor is essential to both computation and performance. Finally, we examine the performance of different random walk models and damping factors in identifying Web communities from pure graph topology. The empirical results demonstrate that our random walk models are sufficiently flexible to capture different levels of relationships in the link topology to achieve significant performance in Web community identification. This provides a practical understanding how various random models behave in Web community identification.

2 Background

Before analyzing our specific models, we briefly review related work on identifying Web communities in general.

The problem of identifying Web communities is clearly related to the more fundamental problem of graph partitioning. For general graph partitioning, one can often resort to straightforward principles such as unrestricted minimal cut, or the dual principle of maximum flow. However, the graphs used by most such techniques are undirected, and therefore they ignore the directionality information encoded in Web hyperlinks [4,8]. Another simple approach is to extract similarity measurements between neighboring vertices (Web pages) directly from the link structure to perform a generic clustering method [9]. However, the similarity should be measured from the *global* structure of the graph. A more global approach to Web graph clustering suggests, therefore, that some sort of aggregate similarity measure be used, such as those based on the spectrum of the connectivity matrix. For *undirected* graph clustering, a common suggestion is to partition by performing a singular value decomposition (SVD) on W [13]. However again, the connectivity matrix W is *not* symmetric.

By considering the directed links of Web pages, Kleinberg showed that the HITS ranking algorithm [10] converges to a spectral method that uses the principle eigenvectors of $W^T W$ and WW^T—the final weight scores for the authorities and hubs. Later, it was observed that this technique can in fact be used to identify web communities, where Web pages with highest authority and hub scores are used to define the core of a community [6]. However, one can see that this approach reduces to SVD on an *undirected* graph weight matrices WW^T and

$W^T W$. In fact, this approach suffers from two drawbacks: first, a straightforward graph partition method based on simply computing the principle eigenvectors is not very effective in general; and second, the directed hyperlink information is significantly diminished through the symmetric transformations. Regarding the first drawback, a more appropriate way to solve the graph partitioning problem is to consider it as a *balanced* minimum cut problem, which usually results in more accurate clusters being obtained. Although most versions of the balanced minimum cut are NP-complete, the eigenvectors of graph Laplacians [2] provide a good approximation to this NP-hard problem. The efficiency and effectiveness of such *balanced* spectral clustering methods has been demonstrated in many domains, e.g. [14]. Unfortunately, these methods have only been developed for undirected graphs, and do not consider directionality information.

To address these shortcomings, we require a balanced spectral clustering principle that can take into account the directionality of Web hyperlinks. Recently, a new approach to directed graph clustering has been proposed in [16], which offers a mathematically clean solution to this problem. It minimizes a balanced cut criterion for directed graphs that has a very natural interpretation in a random walk framework. Unfortunately, the work presented in [16] does not address the specific role of random walks in Web graph clustering. A Web graph differs from a general directed graph in that it posesses particular topological properties. It is therefore critical to formulate a proper random walk model that ensures similar pages are grouped into coherent Web communities. In this paper, we analyze two random walk models with their variants that are sufficiently flexible to capture important aspects of Web graph topology, and disclose how walk connectivity is related to page similarity in directed spectral clustering. Below we investigate the performance of these random walk models in comparison with standard models of spectral clustering on undirected graphs [6].

3 Directed Spectral Clustering

To identify Web communities in a Directed Web Graph we employ the efficient spectral clustering technique for directed graphs developed in [16]. The criterion for directed graph partitioning is given by a combinatorial partition criterion that generalizes the normalized cut criterion for undirected graphs [14]. It requires no transformation of the asymmetric adjacency matrix into a symmetric one.

A directed graph $G = (V, E)$ can be associated with a Markov chain defined by a random walk on the graph. The stationary distribution π of this random walk gives a probability of occupancy over a vertex v given infinite time. So given a subset S of vertices in G, we define the probability with which the random walk occupies vertices in S as $P(S) = \sum_{v \in S} \pi(v)$. Let S^c denote the complement of S. Obviously, $P(S) + P(S^c) = 1$. Define the probability with which the random walk jumps to S^c from S as $P(S \to S^c) = \sum_{u \in S, v \in S^c} \pi(u) p(u, v)$. We then consider partitioning the directed graph G into two nonempty subsets S and S^c by minimizing the following

$$\text{cut}(S) = \frac{P(S \to S^c)}{P(S)} + \frac{P(S^c \to S)}{P(S^c)} \tag{1}$$

Intuitively, a good partitioning of a directed graph under this criterion corresponds to a cut such that the probability of escaping from one community to another is small, whereas the probability of remaining in the current community is high. Note that these escape and retention probabilities are measured with respect to a long run of the random walk, so the optimal partition is determined by the global link topology of the graph. Minimizing this objective is NP-hard [2] but an approximation can be efficiently obtained by solving for the eigenvectors of a directed graph Laplacian Δ defined as follows [16]. Let Π denote the diagonal matrix with $\Pi(v,v) = \pi(v)$ for all $v \in V$. Let P denote the transition probability matrix and P^T the transpose of P. Then define

$$\Theta = \frac{\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^T \Pi^{1/2}}{2}. \tag{2}$$

Then, $\Delta = I - \Theta$, where I denotes the identity. The directed spectral clustering algorithm is then to compute the eigenvector Φ of Θ corresponding to the second largest eigenvalue, and then partition the vertex set V of G into two parts according to the sign. In practice, for multiple clusters, it is standard to detect and visualize clusters based on the sorted eigenvalues [1].

4 Random Walks on Digraphs

The random walk model is a free parameter in the directed spectral clustering framework outlined above. Technically, the only requirement is that the transition probabilities of the random walk satisfies the *balance equation* $\pi(v) = \sum_{u \to v} \pi(u) p(u,v)$ where $u \to v \in E$ denotes page v is pointed by page u. However, for the purposes of identifying latent Web communities in a Directed Web Graph, we need to specify an appropriate random walk to ensure that tightly coupled Web pages share a common topic or interest. This provides a practical understanding of different behavior of random walk models in Web clustering.

One-Step Random Walk

The one-step random walk model we examine initially is the *teleporting random walk* model of [12]. Given that the random surfer is currently at a vertex u: (a) with probability ϵ it chooses an outlink uniformly at random and follows the link to the next page; or (b) with probability $1 - \epsilon$ it jumps to a Web page uniformly at random over the entire Web (excluding itself). Here, a damping factor ϵ $(0 < \epsilon < 1)$ is introduced in the case where the current page has no outlink. Such a random walk is guaranteed to converge to a unique stationary distribution which can be computed by numerically solving the balance equation. The transition probability $p_{tele}(u,v)$ between u and v under this model can be written as $p_{tele}(u,v) = \epsilon \frac{w(u,v)}{d^+(u)} + p_\epsilon(u,v)$, where $p_\epsilon(u,v) = w(u,v)/\operatorname{vol} G$ if $d^+(u) = 0$ and $p_\epsilon(u,v) = (1-\epsilon) w(u,v)/\operatorname{vol} G$ if $d^+(u) > 0$; $\operatorname{vol} G = \sum_u (d^+(u) +$

$d^-(u)$). Here $w(u,v)$ is the weight value along each edge; $d^-(v) = \sum_{u \to v} w(u,v)$ and $d^+(v) = \sum_{u \leftarrow v} w(v,u)$ are the *in-degree* and *out-degree* of v.

This random walk makes the simple assumption that similar pages are directly linked. The stationary probability of a Web page corresponds to the frequency that a surfer visits the page following forward links. This can be viewed as an authority effect in the Web page ranking. We refer to this random walk as the one-step authority model (***OneStepA***). Conversely, we can consider another random walk that traverses *backward* along the hyperlinks [3]. This is equivalent to the hub effect, since a good hub page should be able to visit many other related pages. Therefore, we refer to this random walk as the one-step hub model (***OneStepH***).

Two-Step Random Walk

Web pages are "connected" by more than their direct hyperlinks. Intuitively, commonality between two Web pages is revealed by the presence of common co-citation or co-reference pages. The random walk we employ should therefore also consider these implicit connections in Web community identification.

We now consider a *two-step random walk* model motivated by the Hubs and Authorities model in [10]. Assume temporarily that each Web page has inlinks and outlinks. Then, starting from a page u, the random surfer first jumps backward to an adjacent hub vertex h with probability $p^-(u,h) = w(h,u)/d^-(u)$, then it jumps forward to a page v adjacent from h with probability $p^+(h,v) = w(h,v)/d^+(h)$. Then the two-step transition probability $p^A(u,v)$ between two authorities u and v is given by

$$p^A(u,v) = \sum_h p^-(u,h)p^+(h,v) \tag{3}$$

The stationary distribution π^A of this random walk is $\pi^A(u) = d^-(u)/\operatorname{vol} G^-$ where $\operatorname{vol} G^- = \sum_{u \in V} d^-(u)$. This follows from the fact that

$$\begin{aligned}\sum_{u \in V} \pi^A(u)p^A(u,v) &= \sum_{u \in V} \frac{d^-(u)}{\operatorname{vol} G^-} \sum_{h \in V} \frac{w(h,u)w(h,v)}{d^-(u)d^+(h)} \\ &= \frac{1}{\operatorname{vol} G^-} \sum_{h \in V} \frac{w(h,v)}{d^+(h)} \sum_{u \in V} w(h,u) = \frac{d^-(v)}{\operatorname{vol} G^-} = \pi^A(v)\end{aligned}$$

This random walk is performed by treating pages as authorities.

Using the same argument, we can define a two-step random walk by treating pages as hubs. The random walk performs among hubs u and v by first taking a forward step and then a backward step along the edges $u \to a$ and $a \leftarrow v$, yielding the transition probability between hubs

$$p^H(u,v) = \sum_a p^+(u,a)p^-(a,v) \tag{4}$$

Similarly, this random walk between hubs has the stationary distribution $\pi^H(u) = d^+(u)/\operatorname{vol} G^+$. The two-step random walk exploits the co-citation and co-reference effects in the high level Web link topology. The assumption here is that two similar pages should share more common hubs or authorities.[1]

The above two-step random walks require that each Web page has inlinks and outlinks, but this is not always true for real Web graphs. To be able to handle the general case, we propose to combine the two-step random walk with a teleporting step, so that each forward and backward step through an outlink and a inlink has a damping factor. Therefore, to obtain the mixed two-step random walk, simply plug the modified transition probabilities p^- and p^+ into formulas (3) and (4) to modify p^A and p^H among authorities and hubs. In our experiments below we only use the mixed version of the two-step random walks, but for simplicity we just refer to them as ***TwoStepA*** and ***TwoStepH*** respectively. Finally, we consider a convex combination of the two types of two-step random walks that address the hyperlink structure in a more flexible manner $P = \beta P^A + (1-\beta)P^H$, where β is a tuning parameter that controls the different weights of co-citation and co-reference effects. The advantage of this combination is that it can help us determine which effect is dominant in the link structure, based on the results. Or conversely, given some prior knowledge about the levels of link structure, we can set a proper value for β that consistently matches the hyperlink topology.

Spectral Clustering with Random Walks

To partition a Directed Web Graph, we can simply use the adjacency matrix A with unit weights (i.e., $a(u, v) = 1$ when $u \to v$). It is interesting to compare the results of the different random walk models and the symmetrized transformation models in this case. To demonstrate the differences in a simple toy example, we computed the second eigenvectors of Θ formulated as in (2) for both the one-step and two-step random walks on the graph in Figure 1. We set $\epsilon = 0.95$. We also obtain the principal eigenvectors of $A^T A$ and AA^T, corresponding to the symmetrized authority and hub scores mentioned in the Background section [10]. We refer to these symmetrized methods as ***Auth*** and ***Hub*** respectively.

One can partition the directed graph into two clusters by examining the values in the eigenvector thresholding at zero. Pages within an initial grouping can then be partitioned further after the first partitioning [1], and so on. In addition to just partitioning the vertices, however, the eigenvector values can also be used to assign a *weight* or *confidence* that each Web page belongs to its assigned cluster. That is, the greater the eigenvector value at a page, the more likely the page is to belong to the given cluster. We will therefore refer to these values as the *weights* of pages below. We visualize the partitioning by assigning each vertex on a solid line as shown in Figure 1.

[1] We briefly note that [11] uses the stationary distribution proportional to vertex in-degrees to perform a simple ranking method and showed similar derivations of stationary distributions.

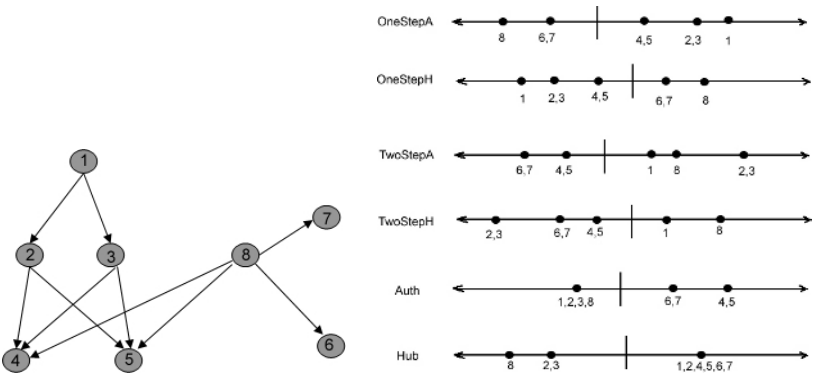


Fig. 1. Left: A toy example of a directed graph. Right: Illustrating partitioning by sorted values. Here, "|" indicates the threshold value (zero) such that vertices on each sides are grouped into separate clusters.

In the toy example, the partitions are the same for OneStepA and OneStepH, which tend to extract highly correlated clusters via direct connections. Moreover the vertices that have large values (e.g., 1 and 8) are also the vertices that have the highest stationary distributions under the random walks. It is known that PageRank ranks Web pages by their stationary distribution, but pages with high stationary probabilities might be of dissimilar topics. However, besides clustering, this method can provide rearranged rankings within each cluster which is very useful to current search engines.

TwoStepA tends to group strong authorities (4, 5, 6, 7) together that are linked by common pages. TwoStepH extracts the hub vertices (1, 8) that link to similar vertices directly and/or indirectly, e.g., vertex 1 points to vertices 4 and 5 after passing 2 and 3. Vertex 8 points to vertices 4 and 5 directly. This random walk tends to group good hubs that link to common pages either implicitly or explicitly. The partition using the symmetrized authority score is similar to TwoStepA, but it does not distinguish among the vertices 1, 2, 3 and 8. The partition using the symmetrized hub score also ignores any differences among the vertices in each group, and is thereby less meaningful.

Random walks are able to effectively capture the differences between direct hyperlink and indirect second order hyperlink topologies that have different co-citation and co-reference patterns in directed spectral clustering. All of these can be exploited to efficiently identify vertex communities via directed spectral clustering.

Table 1. Web graphs statistics

Root queries	vertex num	edge num
1. "waterloo"	2130	4688
2. "movies"+"olympics"	6634	65536
3. "risk analysis"+"bussiness optimization"	3357	10490
4. "differential geometry"+ "parallel computing"	2575	6844
5. "data mining"+"computer vision"	3907	12416
6. "body arts"+"fashion design"	3091	4122

5 Empirical Results

5.1 Experimental Design

We construct Web graphs of varying degrees of difficulty by either building the graph from a single topic query, which results in multiple topics that can be hard to distinguish, or building the graph from multiple queries, which results in a few more easily distinguishable topics. To obtain Web graphs, we first chose some *root queries*, submited these to Google, and retrieved the first t html pages (not including pdf or ps files). For a given query or set of queries, we then combined the retrieved pages as *roots* and perform a one level expansion by adding pages that are linked from or link to the root pages. Finally, we filtered out non-informative links that exist among Web pages as follows. We restrict the number of pages that link to or are pointed to by every root URL to be at most d pages. This operation was first proposed in [10]. We also filter out all *cgi scripts* links. We set t and d equal to 100 and 50 respectively. The collections we finally obtained were relatively sparse graphs. In our experiments, we use several groups of root queries. Their statistics are listed in Table 1. The root queries focus on a variety of interests. Pages retrieved from queries that have significant overlap intuitively should increase the difficulty of Web page clustering.

5.2 Results

Choosing Parameters. Practically, two parameters need to be selected when defining the random walks on the Web graphs: the damping factor ϵ in the one-step and two-step random walks, and the tuning parameter β in the two-step random walks. We test with 2 root queries using the damping factor ϵ set to 0.75, 0.85 and 0.95. Clustering performance is evaluated by counting the correctly classified pages that have the 30 greatest weights among those ranked within top 100 by Google.

Figures 2 plot the confusion matrix values corresponding to the numbers of pages among the 30 with the greatest weight that are classified as "movie" (class 1) and "olympics" (class 2). Ideally, the best result should have corresponding numbers of 30, 0, 0, 30. Since OneStepA and OneStepH give very similar results in this experiment, we only show the results of OneStepA. One can see from these figures that the directed spectral method with OneStepA obtains the best performance when ϵ equals 0.85. Thus, we fix this value for OneStepA in later experiments. For TwoStepA, the results are competitive when ϵ takes value 0.85 and 0.95. Since each result has a better performance for one of the communities, we choose $\epsilon = 0.90$ as an compromise value in the following experiments.

Next, we consider the tuning parameter β that balances between P^A and P^H in the two-step random walk. Figure 3 (left) shows the results when β changes from 1 to 0 in the "movies+olympic" Web graph. Instead of reporting the confusion matrix values in detail, we summarize it by the F *measure*, which can be derived from the confusion matrix as $\frac{2(precision \times recall)}{(precision + recall)}$ where $precision = C_{11}/(C_{11} + C_{21})$ and $recall = C_{11}/(C_{11} + C_{12})$. The Figure shows that the best

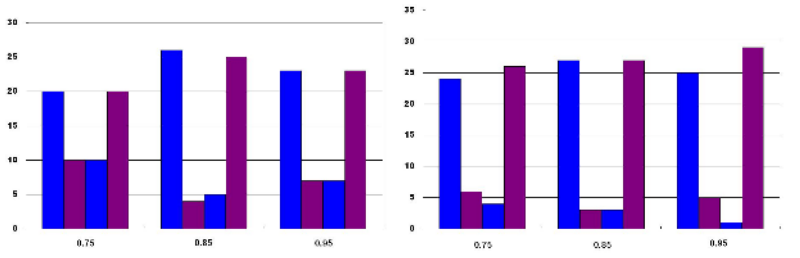


Fig. 2. OneStepA results(left) and TwoStepA results(right). Plot of confusion matrix values $C_{11}, C_{12}, C_{21}, C_{22}$(from left to right of each column block) for $\epsilon = 0.75, 0.85, 0.95$.

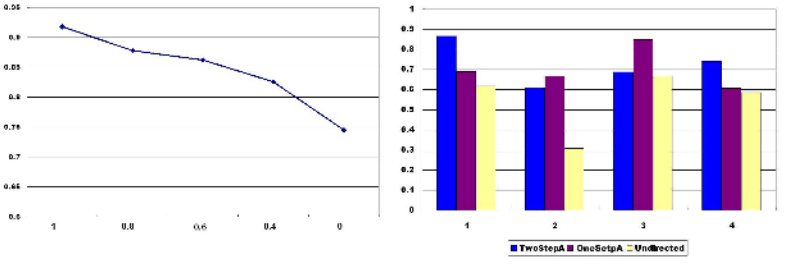


Fig. 3. Left: F scores when β changes in two-step random walk, $\epsilon = 0.90$. Right: F score for 4 binary clustering tasks. Blue: TwoStepA, Red: OneStepA, Yellow: Undirected.

performance is obtained when $\beta = 1$. This means that the Web page similarities are most correctly assessed when the transition matrix is P^A for this Web graph. Not surprisingly, this result is consistent with the ranking methods that consider inlink degree and authority scores from $A^T A$ [6,11]. These studies have already shown that important pages can be found by evaluating their authority scores only. Thus, we set $\beta = 1$ in our following experiments, although one should point out that this is not a globally optimal choice.

Single Broad-Topic Query. Table 2 lists the communities detected by the directed spectral method using OneStepA. For each community, we list the URLs with significant PageRanks. We can visulize that using only hyperlink structure, one can still identify reasonable communities from a Web graph constructed by a single broad topic query. The weights of pages in clusters 1 to 4 are closer to each other than to the pages in other clusters. This discloses that the first 4 clusters are related within a broader scope: they are mainly pages from Waterloo, Canada, including academic institutions, social communities and living. The observation that the weights of clusters 5 and 6 are closer to each other than to the others identifies they are the pages of Waterloo locales in the US. The sub-topics are generalized upward to larger common topics. Cluster 9 identifies the pages from Wikipedia, even though we eliminated links among pages from the same domain.

Table 2. Communities from query "waterloo"

Cluster 1: Pages from universities and schools at Waterloo, Canada	Cluster 2: Pages for the public community service in Waterloo, Canada
www.uwaterloo.ca/ www.wlu.ca/ www.lib.uwaterloo.ca/ www.math.uwaterloo.ca/ www.cs.uwaterloo.ca/ www.wcdsb.edu.on.ca/	www.city.waterloo.on.ca/ www.waterloorecords.com/ www.therecord.com/ www.wpl.ca/ www.wrps.on.ca/ www.oktoberfest.ca/
Cluster 3: Pages for living at Waterloo, Canada	Cluster 4: Pages for life at Waterloo, Canada
www.waterlooinn.com/ www.waterloochamber.org/ www.kwhumane.com/ www.kwsymphony.on.ca/	www.kwymca.org/ www.waterloo.ca/ www.kwag.on.ca/ www.uptownwaterloojazz.ca/ www.kwsc.org/ www.waterloo-biofilter.com/ www.wnhydro.com/
Cluster 5: Pages for Waterloo, Iowa, USA	Cluster 6: Pages for Waterloo in the USA
www.wplwloo.lib.ia.us/waterloo/ www.wcfsymphony.org/ www.waterloocvb.org/ www.waterlooindustries.com/	www.waterloobucks.com/ www.waterloo.k12.ia.us/ www.waterloo.il.us/ www.waterlooindustries.com/
Cluster 7: Pages for Waterloo in Europe	Clusters 8 and 9: Pages for the history of Waterloo from public pages and from wiki
www.trabel.com/waterloo/ waterloo-thebattle.htm/ www.waterloo.org.uk/ www.trabel.com/waterloo/waterloo.htm/ www.napoleonguide.com/ battle_waterloo.htm/ www.waterloo.co.uk/	www.garywill.com/waterloo/ history.htm/ www.bbc.co.uk/history/war/ trafalgar_waterloo/ en.wikipedia.org/wiki/ Battle_of_Waterloo/ en.wikipedia.org/wiki/Waterloo_station/

Multiple Topic Related Queries. We also evaluated clustering performance for 4 Web graphs that are obtained from multiple root queries. We compare the directed spectral methods using one-step random walk and two-step random walks to the undirected method that uses the symmetrized authority scores from A^TA (referred to as the undirected method in the results). This undirected method is more efficient than performing SVD on A^TA in undirected graph clustering which is essentially the method in [6] as been explained in Background. Therefore our comparison is more challenging.

Figure 3–Right shows the clustering results for 4 Web graphs obtained from root queries 3, 4, 5 and 6. Not surprisingly, both of the directed spectral methods outperformed the undirected method in all cases.

Table 3. Pages with the top 10 significant weights for Queries of "computer vision" + "data mining"

Directed spectral method with OneStepA		**Undirected method.**	
URL	Cat	URL	Cat
cmp.felk.cvut.cz/eccv2004/	1	dms.irb.hr/index.php	2
iris.usc.edu/Vision-Notes/bibliography/contents.html	1	www.comp.leeds.ac.uk/vision/	2
www.intel.com/research/mrl/research/opencv/	1	www.comp.leeds.ac.uk/vision/	1
marathon.csee.usf.edu/	1	www.statsoft.com/textbook/stdatmin.html	2
vis-www.cs.umass.edu/	1	lear.inrialpes.fr/people/triggs/events/iccv03/	1
www.cs.cmu.edu/ cil/vision.html	1	dir.groups.yahoo.com/group/datamining2/	2
www.sciencedirect.com/science/journal/10773142	1	www.acv.ac.at/	1
www.cs.cmu.edu/ cil/v-source.html	1	www-ai.ijs.si/SasoDzeroski/RDMBook/	2
iris.usc.edu/Information/Iris-Conferences.html	1	www.autonlab.org/tutorials/	2
homepages.inf.ed.ac.uk/rbf/CVonline/	1	www.cs.columbia.edu/ sal/hpapers/USENIX/ usenix.html	2
itmanagement.webopedia.com/TERM/D/ data_mining.html	2	www.scd.ucar.edu/hps/GROUPS/dm/dm.html	2
www.ncdm.uic.edu/	2	www.kdnuggets.com/	2
www.kdnuggets.com/	2	www.spss.com/	2
www.dmg.org/	2	www.eco.utexas.edu/ norman/BUS.FOR/course.mat/ Alex/	2
www.salforddatamining.com/	2	www.acm.org/sigkdd/	2
www.spss.com/	2	www.infogoal.com/dmc/dmcdwh.htm	2
www.acm.org/sigkdd/	2	www.the-data-mine.com/	2
www.megaputer.com/	2	www.thearling.com/text/dmwhite/dmwhite.htm	2
www.cacs.louisiana.edu/ icdm05/	2	www.ncdm.uic.edu/	2

Table 4. Pages with top 15 significant weights for Queries "movies"+ "olympics"

Directed spectral method with TwoStepA		Undirected method	
URL	Cat	URL	Cat
www.saltlake2002.com/	1	www.fhw.gr/olympics/ancient/	1
www.specialolympics.org/	1	cityguide.aol.com/main.adp	1
www.olympic.org/	1	www.dallasnews.com/sharedcontent/dws/spt/olympics/vitindex.html	1
www.torino2006.it	1	www.baltimoresun.com/sports/olympics/	1
sports.espn.go.com/oly/index	1	www.latimes.com/sports/olympics/	1
www.athens2004.com/athens2004/	1	diveintomark.org/howto/ipod-dvd-ripping-guide/	2
www.perseus.tufts.edu/Olympics/	1	movies.nytimes.com/pages/movies/	2
www.perseus.tufts.edu/Olympics/sports.html	1	news.bbc.co.uk/sport1/hi/other_sports/olympics_2012/default.stm	1
news.bbc.co.uk/sport1/hi/olympics_2004/default.stm	1	www.austin360.com/movies/content/movies/	2
www.nbcolympics.com/	1	www.musicfromthemovies.com/default.asp	2
www.olympics.com.au/	1	sports.yahoo.com/olympics	1
www.fhw.gr/projects/olympics/	1	movies.yahoo.com/mv/upcoming/	2
www.london2012.org/	1	www.fairolympics.org/en/	2
en.beijing-2008.org/	1	cbs.sportsline.com/u/olympics/2002/	1
www.imdb.com/	2	www.imdb.com/	2
us.imdb.com/	2	us.imdb.com/	2
www.imdb.com/search	2	rogerebert.suntimes.com/	2
movies.go.com/	2	www.lordoftherings.net/	2
www.usatoday.com/life/movies/front.htm	2	www.allmovie.com/	2
movies.aol.com/	2	www.rottentomatoes.com/	2
movies.yahoo.com/	2	www.infonegocio.com/xeron/bruno/olympics.html	1
movies.guide.real.com	2	www.brainpop.com/	2
www.rottentomatoes.com/	2	www.foxmovies.com/	2
www.hollywood.com/	2	www.hollywood.com/	2
www.boxofficemojo.com/	2	www.reel.com/	2
www.movieflix.com/	2	www.perseus.tufts.edu/Olympics/	1
www.ifilm.com/	2	www.ucmp.berkeley.edu/geology/tectonics.html	1

We also show some of the clustering results by listing the highly ranked URLs with the most significant weights in corresponding communities in Tables 3 and 4."Cat" denotes the ture category for each URL. Once again, we can see that the directed spectral methods work better than the undirected method by tending to group pages more correctly. For example, in Table 3, the pages correctly clustered in the data mining community are about major conferences, term explanations, and companies in data mining. In Table 4, we see in the olympics community, multiple homepages from the olympic game hosts were obtained. Although these pages do not have hyperlinks between them, they all are pointed to by the Olympic Games organization (olympic.org). Thus, the two-step random walk was able to detect their similarity by identifying a common hub. Similar observations can be made about the pages classified in the movies community. In each of these tasks, the undirected method failed to identify pages from same communities, and tended to mix pages from the different communities.

6 Conclusion

To automatically identify Web communities from hyperlink topology, we addressed a key component in directed spectral clustering: the random walk model that should be used to infer relationships between Web pages. In addition to one-step random walks, we also proposed variations of two-step random walk models that can detect higher order similarities between pages. The linear combination of two-step random walks suggests a practical approach to inferring the relationship between link structure and topic similarity by inspecting the clustering results. The experiments show that the different random walk models

can capture different relationships based on the hyperlink topology in directed spectral method.

Acknowledgments

Thanks to Dengyong Zhou for helpful discussions. Work supported by the Alberta Ingenuity Centre for Machine Learning, NSERC, and MITACS.

References

1. S. Chakrabarti, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Spectral filtering for resource discovery. In *ACM SIGIR workshop on Hypertext Information Retrieval on the Web*, 1998.
2. F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
3. C. Ding, X. He, P. Husbands, H. Zha, and H. Simon. PageRank, HITS and a unified framework for link analysis. Technical report, LBNL, 2002.
4. G. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *ACM SIGKDD*, 2000.
5. G. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 2002.
6. D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *UK Conference on Hypertext*, 1998.
7. M. Henzinger. Hyperlink analysis for the web. In *IEEE Internet Computing*, 2001.
8. H. Ino, M. Kudo, and A. Nakamura. Partitioning of web graphs by community topology. In *WWW*, 2005.
9. M. Kessler. Bibliographic coupling between scientific papers. In *American Documentation*, 1963.
10. J. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 1999.
11. R. Lempel and S. Moran. The stochastic approach for link-structure analysis (salsa) and the tkc effect. In *WWW*, pages 387–401, 2000.
12. L. Page, S. Brin, R. Motwani, and T.Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
13. P. Perona and W. Freeman. A factorization approach to grouping. In *ECCV*, pages 655–670, 1998.
14. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 2000.
15. H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *JASIS*, 1973.
16. D. Zhou, J. Huang, and B. Scholkopf. Learning from labeled and unlabeled data on a directed graph. In *ICML*, 2005.

Information Marginalization on Subgraphs

Jiayuan Huang[1,2], Tingshao Zhu[2], Russell Greiner[2],
Dengyong Zhou[3], and Dale Schuurmans[2]

[1] University of Waterloo, Waterloo, Canada
[2] University of Alberta, Edmonton, Canada
[3] NEC Laboratories America, Inc.

Abstract. Real-world data often involves objects that exhibit multiple relationships; for example, 'papers' and 'authors' exhibit both paper-author interactions and paper-paper citation relationships. A typical learning problem requires one to make inferences about a subclass of objects (e.g. 'papers'), while using the remaining objects and relations to provide relevant information. We present a simple, unified mechanism for incorporating information from multiple object types and relations when learning on a targeted subset. In this scheme, all sources of relevant information are marginalized onto the target subclass via random walks. We show that marginalized random walks can be used as a general technique for combining multiple sources of information in relational data. With this approach, we formulate new algorithms for transduction and ranking in relational data, and quantify the performance of new schemes on real world data—achieving good results in many problems.

1 Introduction

Currently, most text classification and clustering algorithms base their inference on the co-occurrence statistics of terms appearing in documents by representing document-term relations via a *bipartite graph*. Many algorithms have been developed for clustering in bipartite graphs, i.e., [9,2,8,4,3]. The underlying intuition behind these approaches is that the similarities among one type of object can be used by the other type of object for clustering.

One obvious limitation of existing co-clustering methods is that they can only deal with two types of data objects, whereas most data sets contain more than two types of objects. For example, in a paper classification task, beyond the bipartite interaction between papers and authors, it is also useful to consider other sources of relevant information, such as the conferences where the papers were published. Such additional paper-conference information could help enhance the classification performance. In this case, one could construct a *tripartite graph* $G = (\langle A, B, C \rangle, E)$, where the vertex sets correspond to authors, papers, and conferences respectively, and E is the set of edges, as shown in Figure 1–left. One could consider addressing the problem of higher-order-partite graphs in a trivial manner by applying co-clustering on each pair of object types; that is, apply a co-clustering method on A, B, and then on B, C individually. However

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 199–210, 2006.
© Springer-Verlag Berlin Heidelberg 2006

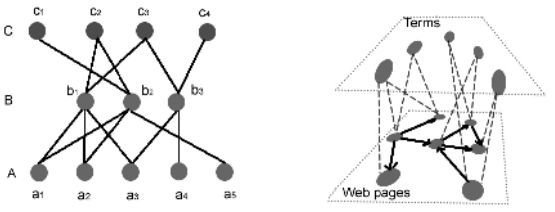


Fig. 1. Left: A tripartite graph. **Right**: A graph of Web pages and terms.

it is hard to ensure the solutions are consistent at the intersection on B. [1] and [5] proposed methods for solving clustering with interactive relationships among multiple object types using ideas from information theory and spectral graph clustering, but they needed to employ sophisticated and computationally expensive methods like semidefinite programming to keep the partitions consistent.

Beyond tripartite clustering, more complex scenarios arise when one considers relationships among data objects of the same type. Previous work on clustering with bipartite and *k-partite* graphs has, for the most part, not taken the relationships *between* objects of the same type into account. Obviously, such information is simply ignored if we present the data as a k-partite graph.

Moving beyond documents and terms, if one considers clustering Web pages, it is clear that the bipartite graph information between Web pages and terms ignores significant relevant information encoded in the hyperlink structure [7,6,10]. When clustering Web pages, it seems clear that both hyperlink structure and term co-occurrence are relevant sources of useful information that one would like to take account of in a unified way. Ideally, one would just model the relationships between Web pages and terms as vertices in a graph like the one shown in Figure 1–right. To the best of our knowledge, clustering in data sets with multiple object types, and multiple relationships between objects of various types has not been well studied in the graph partitioning literature.

In this paper, we propose a simple, unified mechanism for learning in complex scenarios, like the ones shown above, in a graph based approach. We model all data objects as vertices in a graph; e.g., a k-partite graph or a mixed graph as shown in Figure 1–right. The graph based representation allows a simple mechanism for propagating useful information globally throughout a large database of objects: based on the graph, a natural random walk model can be defined that communicates information in a Markov chain. To summarize information from multiple object types and relations when making inferences about one object type, we marginalize the transition probability of the random walk onto the target subset, based on the transition probability of the *induced subgraph* and the transition probability between the subset and its *complement*. In this way, we obtain a valid, new random walk model on the induced subgraph that summarizes all external sources of relevant information. Two objects in the target subgraph that share a lot of common external information will be highly linked in the induced random walk, even if they share no direct links in the induced

subgraph. Once a valid random walk model has been defined, one can derive algorithms for transductive classification, clustering and ranking, by performing random walks over a Markov Chain [10]. The idea of marginalization is a simple and elegant way of dealing with many types of complex scenarios uniformly. Interestingly, when dealing with graphs that happen to be bipartite, the clustering method implied by marginalization is equivalent to the spectral co-clustering method proposed in [9,2]. That is, we recover prominent bipartite graph based inference methods as a special case.

Furthermore, the marginalization idea can be extended to solve more general and interesting types of inference problems on graphs than having been commonly studied in graph partitoning. Consider the problem of clustering the set of blog pages on the Web. In a conventional approach, one could use the induced subgraph on blog pages (namely the subgraph of all the blog pages and their hyperlink structure) to classify the blog pages with respect to their common topics. However, the difficulty with this approach is that there is not much information in the hyperlinks between blog pages, as the owners of the blogs typically do not add links to other blogs if they do not know each other. Therefore, the information obtained directly from the subgraph is not enough to identify blogs of common interest. It therefore makes sense to explore the hyperlinks that *connect blog pages to other general web pages*. For example, people who are interested in computer programming might add a link from their blogs to the page "the art of computer programming" created by Donald Knuth. Although the blogs themselves may have only a few direct links, the blogs can still be clustered into identifiable communities by detecting the pages of common interest linked from the blogs. The scheme we propos can fully exploit all sources of relevant information in a graph of heterogeneous objects to achieve better performance on the target subset.

2 Preliminaries

A *bipartite graph* $G = (\langle A, B \rangle, E)$ is a graph that consists of two *disjoint* sets of vertices, A and B, and a set of edges, E, between A and B. (Typically, the two sets represent different objects, e.g. documents and terms.) Each edge (a, b) is associated with a similarity weight $w(a, b)$. One can generalize bipartite graphs to higher order *k-partite graphs*, whose vertices are divided into k disjoint sets.

Given an undirected graph, a natural random walk can be defined by the transition probability $p : V \times V \rightarrow \Re^{\geq 0}$ such that $p(a, b) = w(a, b)/d(a)$ for all $(a, b) \in E$, where $d(a) = \sum_b w(a, b)$. If the edges have directions, then p is defined by $p(u, v) = w(u, v)/d^+(u)$ for all $(u, v) \in E$ and 0 otherwise, where $d^+(u) = \sum_{u \rightarrow v} w(u, v)$. The random walk on a connected graph has unique *stationary distribution* π that satisfies the *balance equation* $\pi p = \pi$.

Given a general graph $G = (V, E)$ (directed or undirected), and a subset $S \subset V$ of the vertices, the *induced subgraph* with respect to S is the subset V of vertices of G together with any edges whose endpoints are both in V.

3 Learning on a Ergodic Markov Chain

Before presenting our approach in detail, we briefly review related techniques for clustering and transductive learning in graphs involved with Markov chain properties of natural random walks [10]. A graph $G = (V, E)$ can be associated with a Markov chain defined via a random walk on the graph. The stationary distribution of this random walk gives a probability distribution over the vertices v in the graph.

Let $\mathcal{H}(V)$ denote the space of partitions of the vertices V, in that each $f \in \mathcal{H}(V)$ maps each $v \in V$ into real values between -1 and 1. We assume that most linked vertices as similar—that is, belong to the same class. This means, in particular, that all vertices from a densely linked subgraph are likely to have the same label. This motivates us to define the functional

$$\Omega(f) := \frac{1}{2} \sum_{[u,v] \in E} \pi(u) p(u,v) \left(\frac{f(u)}{\sqrt{\pi(u)}} - \frac{f(v)}{\sqrt{\pi(v)}} \right)^2$$

that sums the weighted variation of a function on each edge of the directed graph. The labels are smoothed over the entire graph by minimizing the variation.

There is a equivalent way to express $\Omega(f)$. Let Π denote the diagonal matrix with $\Pi(v,v) = \pi(v)$ for all $v \in V$; let P denote the transition probability matrix; and let P^T the transpose of P. Then

$$\Theta = \frac{\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^T \Pi^{1/2}}{2}.$$

Using I for the identity matrix, it can be proved that $\Omega(f) = f^T(I - \Theta)f$. The functional $\Omega(f)$ can also be derived with respect to a normalized cut criterion that generalizes the standard spectral clustering criterion to directed graphs [10].

4 Marginalized Random Walks on a Subgraph

We can model many versions of graph-based inference problems as learning on an induced subgraph. Typical learning tasks in this setting are classification and clustering on a target subset, where one would like to utilize not only the original structure of the subgraph, but also the global structure and the interactions between the subgraph and its complement. To propagate the information needed to perform these tasks, the graph based approach depends upon a random walk model to communicate the relevant information globally throughout the graph. In the case where the inference problem is to be localized on a focused subset of the graph, we need a new random walk model that communicates the sources of relevant information to the subset. With an appropriate *marginalized* random walk model, we can then derive principled techniques for transductive classification, clustering and ranking.

Given a graph $G = (V, E)$ (either directed or undirected), and a subset of vertices $A \subset V$, we are interested in performing a learning task in A, e.g.,

learning a classification of A's vertices. We let A^c denote the complement of A. For example, in the blog example where A is the set of blog pages we want to classify based on topic, A^c is the set of non-blog Web pages that have connections to the blog pages. In the example of a tripartite graph for a citation network including papers, authors and conferences, A is the set of papers and A^c includes all the authors of the papers and the conferences.

Typically, the transition probability P of a natural random walk model on the graph can be written as in Section 2. Here we can equivalently rewrite the transition probability in a blockwise form with respect to A and A^c

$$P = \begin{pmatrix} P_{AA} & P_{AA^c} \\ P_{A^cA} & P_{A^cA^c} \end{pmatrix}$$

where P_{AA^c} denotes the transition probability between vertices in A and A^c, etc.

One could attempt to perform classification in A based only on P_{AA}, by applying the framework reviewed in Section 3. However this ignores the information that connects A and A^c, which could be significant. A extreme case is that when we have no interactive relationships in either A or A^c but only P_{AA^c} and P_{A^cA}; that is, a bipartite graph (when edges between A and A^c are undirected). We will see later in Section 4.1 that co-clustering methods utilize P_{AA^c} and P_{A^cA} in an undirected case. Now our goal is to define a new random walk in A incorporating all relevant information.

Given a vertex u in A, we first assume it has outlinks to a vertex v in A and a vertex v_c in A^c. The random walk has the following two options starting from u: it can follow the outlink to v (and so stay within A), or to v_c (and so leave A). If it stays in A, the random surfer follows the transition probability P_{AA}. If the random surfer jumps out of A to A^c, its walk will follow the transition probability P_{AA^c}. Once it enters A^c, there is a non-zero chance it will take any number of steps in A^c before possibly returning to A. Therefore, we can write the transition probability between u and v in A, if the surfer re-entered A after transiting from A to A^c and back to A as,

$$P_{out} = P_{AA^c}\left(I + \sum_{i=1}^{n\to\infty} P_{A^cA^c}^i\right) P_{A^cA} = P_{AA^c}(I - P_{A^cA^c})^{-1}P_{A^cA}$$

In addition, define $P_{in} = P_{AA}$ if the surfer stays within A. Combining these two transition models yields a new random walk on the *subgraph* A, whose transition probability P_{AA}^* is given by

$$P_{AA}^* = P_{in} + P_{out}$$

To ensure P_{out} and P_{AA}^* are well defined, we assume P is ergodic. We then have the following claims.

Claim. $I - P_{A^cA^c}$ is invertible.

Proof (of claim). Assume $I - P_{A^cA^c}$ is singular. Then $(I - P_{A^cA^c})x = 0$ has a non-trivial solution $x = P_{A^cA^c}x$. Taking norms, we have $|x| = |P_{A^cA^c}x| \leq$

$|P_{A^cA^c}|\,|x| \;<\; |x|$. The last inequality follows because the row sum of $P_{A^cA^c}$ is less than 1. Contradiction.

Claim. P^*_{AA} is a valid transition probability; i.e. the sum of each row equals 1.

Proof. Consider the ways a random surfer can start from a vertex u in A and return to another vertex v in A. In the first step, u has two choices, either follow links in A or jump out of A to A^c. If it stays in A, the transition probability is P_{in}. If it jumps out of A, then the surfer has an infinite number of paths lenghts that stay in A^c, before (possibly) returning to A. Here, P_{out} is the probability of transiting from u to v via A^c and P_{in} is the transition probability from u to v without entering A^c. Thus the sum of these two disjoint transition probabilities is a valid transition probability.

We let P^*_{AA} denote the new transition probability on A by marginalizing the random walk on subset A, taking all sources of information into account. The similarity among vertices in A is measured by a combination of the transition probability within A, P_{in}, and the probability of escaping from A to A^c and then returning to A, P_{out}. Therefore, we define a new Markov Chain over the subset of the graph. We can use the functional (3), to produce graph-based algorithms for transductive classification, clustering and ranking on complex graphs:

$$f^* = \arg\min_f \{\Omega(f) \;+\; \mu\,||f - y||^2\}$$

Here $y = \langle\, y_i \,\rangle$ is the partially labeled vector; where each labeled data is either 1 or -1, and $y_i = 0$ for each unlabeled data point. For ranking, we label the *root* data as 1 and the rest as 0. Also, μ is a tuning parameter; where for clustering tasks we set $\mu = 0$ since we do not have any label information.

4.1 Learning with a Bipartite Graph

In this section, we will show that the original spectral co-clustering on a bipartite graph [9,2] can be equivalently interpreted as defining new random walk models on each subset of the bipartite graph in our scheme.

Given a bipartite graph $G = (\langle\, A, B \,\rangle, E)$, where A and B are disjoint subsets of vertices, the transition probability P over G has the following blockwise form:

$$P \;=\; \begin{pmatrix} 0 & P_{AB} \\ P_{BA} & 0 \end{pmatrix}$$

Thus, as in the previous section, we can define new random walk in A and B as

$$P^A = P_{AB}P_{BA}, \tag{1}$$

$$P^B = P_{BA}P_{AB} \tag{2}$$

Intuitively, such random walks can be also understood as a two step random walk, motivated by the Hub and Authority model. We take vertices in B as

the evidence of existing similarities between nodes in A. The similarities are mutually reinforced via the random walk between them, as follows.

First consider the random walk among vertices in A (B will be isomorphic). If the random surfer is currently at vertex $a_i \in A$, it first takes a backward step along edge (a_i, b) to some vertex $b \in B$. Then if b also has an edge connected to a_j, the surfer will visit a_j along the edge (b, a_j).

The two-step transition probability $p^A(a_i, a_j)$ is determined by the surfer taking one backward step and one forward step. Therefore,

$$p^A(a_i, a_j) = \sum_b p(a_i, b)p(b, a_j) = \sum_b \frac{w(a_i, b)w(b, a_j)}{d(a_i)d(b)} \tag{3}$$

which is exactly the same as the P^A obtained in (1).

The stationary distribution π^A of this random walk is

$$\pi^A(a) \;=\; \frac{d(a)}{\operatorname{vol} G_A} \tag{4}$$

where $\operatorname{vol} G_A = \sum_{a \in A} d(a)$. This means

$$\begin{aligned}\sum_{a_i \in A} \pi^A(a_i)p^A(a_i, a_j) &= \sum_{a_i \in A} \frac{d(a_i)}{\operatorname{vol} G_A} \sum_{b \in B} \frac{w(a_i, b)w(b, a_j)}{d(a_i)d(b)} \\ &= \frac{1}{\operatorname{vol} G_A} \sum_{b \in B} \frac{w(b, a_j)}{d(b)} \sum_{a_i \in A} w(a_i, b) = \frac{d(a_j)}{\operatorname{vol} G_A} = \pi^A(a_j)\end{aligned}$$

Similarly, we can define the two step transition process among nodes in B, yielding the transition probability

$$p^B(b_i, b_j) = \sum_a p(b_i, a)p(a, b_j) = \sum_a \frac{w(b_i, a)w(a, b_j)}{d(b_i)d(a)} \tag{5}$$

which corresponds to (2). Moreover, the stationary distribution π^B is

$$\pi^B(b) = \frac{d(b)}{\operatorname{vol} G_B} \tag{6}$$

To obtain classification or clustering results on both subsets simultaneously, we define a smoothness function f over A from (3) that is measured by

$$S_A(f) = \frac{1}{2} \sum_{a_i, a_j} P^A(a_i, a_j)\pi(a_i) \left(\frac{f(a_i)}{\sqrt{\pi(a_i)}} - \frac{f(a_j)}{\sqrt{\pi(a_j)}} \right)^2$$

Similarly, the smoothness function g over B is defined as

$$S_B(g) = \frac{1}{2} \sum_{b_i, b_j} P^B(b_i, b_j)\pi(b_i) \left(\frac{g(b_i)}{\sqrt{\pi(b_i)}} - \frac{g(b_j)}{\sqrt{\pi(b_j)}} \right)^2$$

We can use (3), (4), (5) and (6) to prove that

$$S_A(f) = \frac{1}{\text{vol}\, G_A} f^T \Delta_A f, S_B(g) = \frac{1}{\text{vol}\, G_B} g^T \Delta_B g$$

where

$$\begin{aligned} \Delta_A = I - D_A^{-1/2} W^T D_B^{-1} W D_A^{-1/2} &= I - MM^T \\ \Delta_B = I - D_B^{-1/2} W D_A^{-1} W^T D_B^{-1/2} &= I - M^T M \end{aligned}$$

where $D_A = We$, $D_B = W^T e$ and $M = D_A^{-1/2} W^T D_B^{-1/2}$ using the all-1 vector e. W is the weight matrix between A and B. The solutions for f and g are the eigenvectors of MM^T and $M^T M$ with second largest eigenvalues.

It is known the solution of spectral co-clustering on A and B is the second largest left and right singular vectors of M [9,2]. It is easy to see that from the singular value decomposition, that the non-zero left singular eigenvalues of M are the square roots of the non-zero eigenvalues of MM^T with the same eigenvector space. The eigenvector space of M's right eigenvectors is the same as the one of $M^T M$. Therefore, the two solutions are exactly the same, but with different motivations.

The advantage of having marginalized random walk models on each subset is that we can treat each set individually while using their mutual relationships. As expected, the solution is exactly the same as when we considered the combinatorial cut problem in bipartite graphs. In spectral co-clustering method, the goal is to define a cut criterion for the weight matrix that minimizes the cut over the unmatched edges and maximizes the matched vertices in the subgraphs. Such cuts naturally partition the bipartite graph into two parts in each set. The solution is not clear though if we want different number of partitions on each subset. While using our scheme, we can obtain k-cluster results using the first k eigenvectors of Δ_A and Δ_B. Moreover, as discussed in Section 4, this method can be easily generalized into more complex graphs, which would have been difficult from graph cut perspective.

5 Experiments

In this section, we demonstrate several problem settings that involve data represented in complex graph structures. We evaluate our information marginalization approach by applying it to two datasets; see Sections 5.1 and 5.2.

The first dataset is from WebKB (`www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data`), which includes pages from four universities: Cornell, Texas, Washington and Wisconsin. After removing isolated pages, the Web pages have been manually classified into seven categories: *student, faculty, staff, department, course, project* and *other*. We take advantage of the link structure and page-word relationships for the following two learning tasks.

(1a) Given the link structure of all the pages and the words used in them, discriminate student (course) pages from non-student (non-course) pages. Here, A corresponds to the web pages, and A^c to the words. See Figure 1.

(1b) Given only the link structure, discriminate student pages (labeled as 1) from course pages (labeled as -1). For this task, A corresponds the pages of students and courses, and A^c to the web pages from other classes.

The second dataset is based on CiteSeer (`citeseer.ist.psu.edu/`)—a well-known scientific digital library that catalogues primarily computer and information science literature. We construct our citation networks based on paper-paper and paper-author relationships from CiteSeer. We extract a set of papers P with authors U. Here, we focus on two kinds of ranking.

(2a) Given some papers (i.e., *seed* papers) in P labeled as relevant to a specific topic T, rank the rest of the papers based on their relevance to T. Here, A is P, A^c is U.

(2b) Given some authors (i.e., *seed* authors) in A identified as relevant since they share similar research interests, rank the remaining authors based on how much they share the research interests with these seed authors. A is U, A^c is P.

To build citation networks, we scout ahead following the paper citation and corresponding authors information from the OAI records (`citeseer.ist.psu.edu/oai.html`). We start a crawl from a set of pre-selected authors (i.e., *root authors*), then collect all their papers and the co-authors of these papers. The co-authors are added to a growing set of authors that is used in the next iteration. We repeat this iteration $n = 3$ times to collect a number of related authors and papers. In our experiment, we choose the root authors from two different areas:

Root authors	# Authors	# Papers
"Berhard Scholkopf" + "John Kleinberg"	7156	4979
"Vladimir Vapnik" + "Jianbo Shi"	3048	2097

Therefore, the citation network contains authors with different research subjects, which is more realistic.

5.1 Results: Web Classification

We compare the performance of two algorithms for Web page classification in transductive setting. It is well-known that transductive classification typically outperforms supervised one because it take advantages of unlabled data in the learning procedure. The first transductive algorithm uses our marginalized random walk P^*, and the second one uses hyperlink structure P_{AA} only. We use canonical 0-1 weights over the directed hyperlinks. We set the tuning parameter $\mu = 2.5$ for both algorithms. We increase the size of the labeled data sample at each iteration. The comparison is based on 0/1 classification error, averaged by 20 iterations.

Figures 2 and 3 show the comparison results for problem (1a), and Figure 4, for problem (1b). It is clear that the methods using information marginalization outperforms the one with only the local hyperlink information from subset. Specifically, this implies that the marginalized random walk is able to convey more global information onto the subset, efficiently improving the performance in classification.s

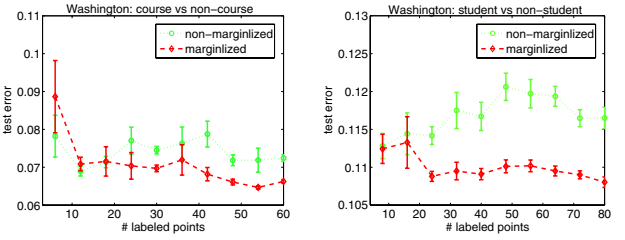


Fig. 2. Classification error on discriminating course pages from non-course pages (left) and student pages from non-student pages (right) from Washington

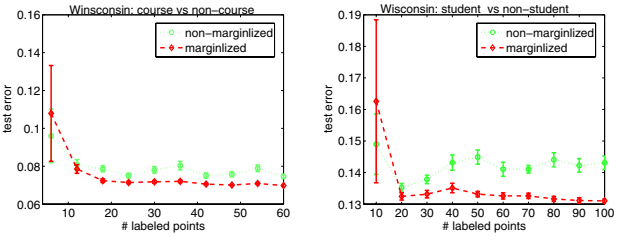


Fig. 3. Classification error on discriminating course pages from non-course pages (left) and student pages from non-student pages (right) from Wisconsin

5.2 Results: Ranking in Citation Networks

For problem (2a), Table 1 shows the top 20 results of paper ranking with respect to the labeled paper "Kernel Principal Component Analysis"; and Table 2 shows the top 10 papers ranked with respect to "Authoritative Sources in a Hyperlinked Environment". We can see that the information maginalization method works better than only using citation links information as the highly ranked papers are closer to the labeled paper in information marginalization scheme. If we only consider citation links, some papers from slightly different domain may be included in the top ranking list because they may have citations with similar papers. With the help of author-paper relationships, the relationship between the labeled paper and other papers become more clear thus lead more accurate ranking results.

For problem (2b),Table 3 lists the ranking results of authors with respect to Vladimir Vapnik in the second citation network. The information from the citation links moves some authors—Chris Burges, Bernhard Scholkopf, Olivier Chapelle and Alex Smola—to higher ranking positions than only using author-paper relationships. The reason is that these authors also have many citation links among their papers that strengthen the similarities with respect to the labeled author.

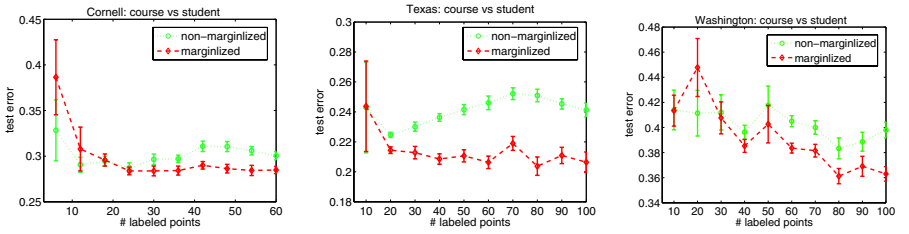


Fig. 4. Classification error on discriminating course pages from student pages

Table 1. Papers Ranked closest to "Kernel Principal Component Analysis"

marginalized random walk	**use only citation links**
title	title
1. Regression Estimation with Support Vector Learning Machines	1. Model Selection for Support Vector Machines
2. Model Selection for Support Vector Machines	2. SV Estimation of a Distribution's Support
3. Support Vector Method for Novelty Detection	3. Support Vector Method for Novelty Detection
4. A Generalized Representer Theorem	4. Optimal Hyperplane Classifier with Adaptive Norm
5. Optimal Hyperplane Classifier with Adaptive Norm	5. Inclusional Theories in Declarative Programming
6. Incorporating Invariances in Support Vector Learning Machines	6. Studies on the Formal Semantics of Pictures
7. Latent Semantic Kernels	7. A Noise-Tolerant Hybrid Model of a Global and a Local Learning Module
8. Sparse Kernel Feature Analysis	8. Latent Semantic Kernels
9. Extracting Support Data for a Given Task	9. Incorporating Invariances in Support Vector Learning Machines
10.Support-Vector Networks	10.A Generalized Representer Theorem
11.Kernel Methods: A Survey of Current Techniques	11.Equivalent Conditions for the Solvability of Nonstandard LQ-Problems with Applications to Partial Differential Equations with Continuous Input-Output Solution Map
12.A Training Algorithm for Optimal Margin Classifiers	12.Hyperbolic Conservation Laws with a Moving Source
13.Improving the Accuracy and Speed of Support Vector Machines	13.Extracting Support Data for a Given Task
14.The Connection between Regularization Operators and Support Vector Kernels	14.Support-Vector Networks
15.Generalization Performance of Regularization Networks and Support Vector Machines	15.On Molecular Approximation Algorithms for NP Optimization Problems
16.Statistical Learning and Kernel Methods	16.Kernel Methods:A Survey of Current Techniques
17.The Kernel Trick for Distances	17.CPU Management for UNIX-based MPEG Video Applications
18.On a Kernel-based Method for Pattern "Recognition," "Regression," "Approximation"	18.Efficient Lossless Compression of Trees and Graphs
19.Advances in Kernel Methods - Support Vector Learning	19.A Precise Semantics For Vague Diagrams
20.Estimating the Support of a High-Dimensional Distribution	20.Redescription, Information And Access

Table 2. Papers Ranked closest to "Authoritative Sources in a Hyperlinked Environment"

marginalized random walk	**use only citation links**
title	title
1. Fast Monte-Carlo Algorithms for finding low-rank approximations	1. volutionary Strategies For Solving Frustrated Problems
2. Evolutionary Strategies For Solving Frustrated Problems	2. Fast Monte-Carlo Algorithms for finding low-rank approximations
3. The Anatomy of a Large-Scale Hypertextual Web Search Engine	3. Reconstruction From The Multi-Component Am-Fm Image
4. Latent Semantic Indexing: A Probabilistic Analysis	4. The Anatomy of a Large-Scale Hypertextual Web Search Engine
5. Challenges in Web Search Engines	5. Latent Semantic Indexing: A Probabilistic Analysis
6. How to Personalize the Web	6. Learning Decision Strategies with Genetic Algorithms
7. Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents	7. A Model for Sequence Databases
8. The PageRank Citation Ranking: Bringing Order to the Web	8. Semantically Driven Automatic Hyperlinking
9. New Results for Online Page Replication	9. Applications of a Web Query Language
10Searching the Web: General and Scientific Information Access	10Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents

Table 3. Author ranking result in network 2

marginalized	**only author-paper relationships**	**marginalized**	**only author-paper relationships**
name	name	name	name
1.Chris Burges	1.Sayan Mukherjee	11.Mark Stitson	11.Vladimir Vovk
2.Bernhard E.Boser	2.Chris Burges	12.Alex Gammerman	12.Alex Gammerman
3.Isabelle M. Guyon	3.Bernhard E. Boser	13.Vladimir Vovk	13.Mark Stitson
4.Sayan Mukherjee	4.Isabelle M.Guyon	14.Chris Watkins	14.Klaus-Robert Muller
5.Donghui Wu	5.Donghui Wu	15.Partha Niyogi	15.Federico Girosi
6.Bernhard Scholkopf	6.Steven E.Golowich	16.Olivier Chapelle	16.Koh.Sung
7.Heinrich H.Bulthoff	7.Volker Blanz	17.Alex Smola	17.Partha Niyogi
8.Thomas Vetter	8.Bernhard Scholkopf	18.Adnan Aziz	18.Jason Weston
9.Volker Blanz	9.Thomas Vetter	19.Jason Weston	19.Olivier Chapelle
10.Steven Golowich	10.Chris Watkins	20.Koh.Sung	20.Alex Smola

6 Conclusions

We have proposed a unified mechanism for incorporating information from multiple object types and relations when making inferences about a targeted subset. Our technique can be applied to learning problems with data embedded in complex graphs. We quantify the performance of our new schemes on two real world relational data and achieve good results in challenging inference problems. Future work will deeply explore more interesting applications of this method.

Acknowledgments

Work supported by Alberta Ingenuity (AICML), NSERC, and MITACS.

References

1. R. Bekkerman, E. El-Yaniv, and A. McCallum. Multiway distributional clustering via pairwise interactions. In *ICML*, 2005.
2. I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioing. In *KDD*, 2001.
3. I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *KDD*, 2003.
4. Ran El-Yaniv and Oren Souroujon. Iterative double clustering for unsupervised and semi-supervised learning. In *ECML*, 2001.
5. B. Gao, T. Liu, X. Zheng, Q. Cheng, and W. Ma. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In *KDD*, 2005.
6. J. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46, 1999.
7. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford, 1998.
8. N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings 37th Allerton Conference*, 1999.
9. H. Zhang, X. He, C. Ding, and M. Gu. Bipartite graph partitioning and data clustering. In *Proceedings of ACM CIKM 2001*, 2001.
10. D. Zhou, J. Huang, and B. Scholkopf. Learning from labeled and unlabeled data on a directed graph. In *ICML*, 2005.

Why Does Subsequence Time-Series Clustering Produce Sine Waves?

Tsuyoshi Idé

IBM Research, Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato, 242-8502 Kanagawa, Japan
goodidea@jp.ibm.com

Abstract. The data mining and machine learning communities were surprised when Keogh et al. (2003) pointed out that the k-means cluster centers in subsequence time-series clustering become sinusoidal pseudo-patterns for almost all kinds of input time-series data. Understanding this mechanism is an important open problem in data mining. Our new theoretical approach (based on spectral clustering and translational symmetry) explains why the cluster centers of k-means naturally tend to form sinusoidal patterns.

1 Introduction

Subsequence time-series clustering (STSC) is one of the best-known pattern discovery techniques from time series data. In STSC, time series data is represented as a set of subsequence vectors generated using the sliding window (SW) technique (see Fig. 1 (a)), and the generated subsequences are grouped, typically using the k-means clustering technique. The cluster centers (the mean vectors of the cluster members) are thought of as representative patterns of the time series.

STSC-based stream mining methods enjoyed popularity until a surprising fact was discovered in 2003 [8]: k-means STSC is "meaningless" as a pattern discovery technique in that the resultant cluster centers tend to form sinusoidal pseudo-patterns almost independently of the input time series.

For clarity, we reproduced the result of Ref. [8]. Figure 2 (a) shows the k-means cluster centers calculated for the time series in Fig. 1[1]. We set the number of clusters and the size of the SW to be $k = 3$ and $w = 128$, respectively. It is surprising that we have sinusoidal patterns in Fig. 2 (a), which are not at all similar to the original patterns in the data. Close inspection shows that the three sinusoids have the same wavelength of w, separated by a phase of $2\pi/3$.

To date, little effort has been made to theoretically pinpoint the origin of the sinusoidal pseudo-patterns, or the *sinusoid effect*. Empirical studies are substantially the only way to validate the attempts to improve STSC. It seems that the lack of theoretical understanding is causing a lack of progress in this area.

[1] A long time series (an example segment is shown in Fig. 1 (a)) was made by concatenating 90 random instances of the Cylinder, Bell, and Funnel (CBF) patterns, whose example instances are shown in Fig. 1 (b).

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 211–222, 2006.
© Springer-Verlag Berlin Heidelberg 2006

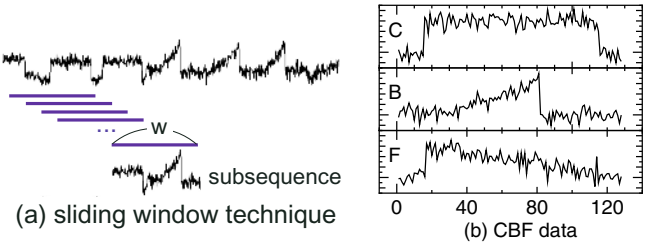


Fig. 1. (a) Sliding window technique and example segment of the concatenated CBF data. (b) Instances of the Cylinder(C)-Bell(B)-Funnel(F) data. There are 30 random instances for each.

This is a theoretical paper. We theoretically show that the SW-based k-means STSC introduces a mathematical artifact into the data, and, unexpectedly, that the artifact is so strong that the resulting cluster centers are dominated by it, irrespective of the details of the data. To the best of the author's knowledge, this is the first work that succeeds in theoretically explaining the sinusoid effect.

The layout of this paper is as follows. In Section 2, we summarize the sinusoid effects and point out a connection to spectral clustering. In Section 3, we present a new theoretical model for time series, which enables us to easily analyze symmetry properties hidden within the problem. In Section 4, we point out that k-means cluster centers can be found by directly solving an eigen equation. In Section 5, we explicitly show why the cluster centers in STSC become sinusoids. In Section 6, we validate our formulation using standard data sets. In the final section, we summarize the paper.

2 Sinusoid Effect in Spectral Clustering

Recently, spectral techniques have attracted great attention as a powerful method for clustering. Some authors [10,9,2,3] have shown the theoretical connection between k-means and certain eigen problems. One interesting question here is whether or not the sinusoid effect is observed in spectral formulations of STSC. Experimentally, it seems that the answer is yes [6]. Specifically, if we think of subsequences generated from a time series as column vectors, and define a matrix H by arranging the vectors as columns, the resulting left singular vectors of H will form sinusoids. We show in Fig. 2 (b) the top three left singular vectors calculated for the same concatenated CBF data. We see that the first ($\boldsymbol{u}^{(1)}$) and the second ($\boldsymbol{u}^{(2)}$) ones are sine waves with wavelength of w, showing clear similarities to Fig. 2 (a).

To summarize these observations in the CBF data:

Observation 1. *The cluster centers of k-means STSC are well approximated by sinusoids with a wavelength of w. While the additive phases are unpredictable, each sinusoid is separated by a phase of integer multiples of $2\pi/k$.*

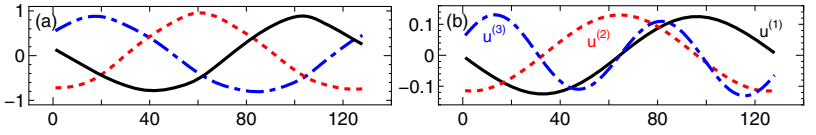


Fig. 2. (a) The k-means cluster centers ($k = 3, w = 128$). (b) The top three feature vectors by SVD ($w = 128$).

Observation 2. *The left singular vectors of the subsequence matrix* H *are well approximated by sinusoids. The top few singular vectors have the wavelength of* w.

These observations suggest that the k-means and singular value decomposition (SVD) of H have a common mathematical structure, and the commonality is the origin of the sinusoid effect. Encouraged by this, we will elucidate the sinusoid effect (1) by reducing the k-means task to that of spectral clustering, and (2) by focusing on the translational symmetry of the problem. In (1), we will introduce a new formulation which directly seeks the cluster centers, instead of the standard formulation based on membership indicators.

3 Preliminaries

3.1 Lattice Model for Time Series Analysis

We define a time series $\mathit{\Gamma}$ as an ordered set of n real-valued variables $x_1, x_2, ..., x_n$. Given a $\mathit{\Gamma}$, a subsequence s_p of length $w \leq n$ is defined by $(x_p, x_{p+1}, ..., x_{p+w-1})$. A subsequence s_p can be viewed as a w-dimensional column vector $\boldsymbol{s}_p$. In STSC, the $\boldsymbol{s}_p$s are thought of as independent vectorial data objects. We focus on SW-based STSC with unit step size and a fixed window size of w in this paper. The number of clusters is represented by k. All vectors are column vectors hereafter.

Any time series $\mathit{\Gamma}$ can be represented as a vector $\boldsymbol{\mathit{\Gamma}}$ in an n-dimensional space. Consider a vector space $\mathcal{H}_0$ spanned by orthonormal bases $\{\boldsymbol{e}_1, ..., \boldsymbol{e}_n\}$, and attach each base $\boldsymbol{e}_l$ to each time point l. Due to the orthonormality, $\boldsymbol{\mathit{\Gamma}}$ can be written as

$$\boldsymbol{\mathit{\Gamma}} = \sum_{l=1}^{n} x_l \boldsymbol{e}_l \tag{1}$$

with $x_l = \boldsymbol{e}_l^{\mathrm{T}} \boldsymbol{\mathit{\Gamma}}$, where the superscript T represents transpose. We call this expression the site-representation (SR) because we can think of our model as the one where each weight x_l is associated with each lattice point or *site* of a one-dimensional lattice having n lattice points.

3.2 Linear Operators in $\mathcal{H}_0$

Let $\mathcal{L}$ be the set of linear operators which transforms a vector in $\mathcal{H}_0$ into another vector. We distinguish the operators by using $\hat{}$ hereafter. By definition, $\forall \hat{o} \in \mathcal{L}$

can be written as a matrix. In particular, it can be written with outer products of the bases in the SR as

$$\hat{o} = \sum_{l,l'=1}^{n} o_{l,l'} \boldsymbol{e}_l \boldsymbol{e}_{l'}^{\mathrm{T}}. \tag{2}$$

The translation operator $\hat{\tau}(l)$

$$\hat{\tau}(l) \equiv \sum_{l'=1}^{n} \boldsymbol{e}_{l'+l} \boldsymbol{e}_{l'}^{\mathrm{T}} \tag{3}$$

is of particular importance. It is easy to verify $\hat{\tau}(l)\boldsymbol{e}_m = \boldsymbol{e}_{m+l}$ and $\boldsymbol{e}_m^{\mathrm{T}}\tau(l) = \boldsymbol{e}_{m-l}{}^{\mathrm{T}}$. The latter suggests

$$\hat{\tau}(l)^{\mathrm{T}} = \hat{\tau}(-l). \tag{4}$$

Hereafter, we assume the *periodic boundary condition* (PBC) to satisfy $\forall l, \boldsymbol{e}_{l+n} = \boldsymbol{e}_l$. As long as $n \gg 1$, the discrepancies due to this artificial condition will be negligible.

3.3 Discrete Fourier Transformation

Consider a subspace $\mathcal{H}$ spanned by $\{\boldsymbol{e}_1, ..., \boldsymbol{e}_w\} \subseteq \mathcal{H}_0$. Here we do not assume the periodicity of w in $\mathcal{H}$. For example, $\boldsymbol{e}_1 \neq \boldsymbol{e}_{1+w}$ unless $w = n$.

We define an orthogonal transformation from the site basis into the Fourier basis as

$$\boldsymbol{f}_q = \frac{1}{\sqrt{w}} \sum_{l=1}^{w} \mathrm{e}^{\mathrm{i}f_q(l-l_0)} \boldsymbol{e}_l \; ; \quad \boldsymbol{e}_l = \frac{1}{\sqrt{w}} \sum_{q \in \mathcal{D}_f} \mathrm{e}^{-\mathrm{i}f_q(l-l_0)} \boldsymbol{f}_q, \tag{5}$$

where l_0 is an arbitrary real number. For simplicity, we abuse the notation f_q to represent $2\pi q/w$, which we call the *wave number*. The subscript q runs over $\mathcal{D}_f = \{-\frac{w-1}{2}, ..., 0, 1, .., \frac{w-1}{2}\}$ when w is odd, and over $\{-\frac{w}{2}+1, ..., 0, 1, .., \frac{w}{2}\}$ when w is even. It is straightforward to show $\boldsymbol{f}_q^{\mathrm{T}} \boldsymbol{f}_{q'} = \delta_{q',q}$, and thus, $\{\boldsymbol{f}_q\}$ forms a complete set in $\mathcal{H}$.

For $\forall \boldsymbol{\gamma} \in \mathcal{H}$, the discrete Fourier transformation (DFT) is defined as

$$\boldsymbol{\gamma} = \sum_{q \in \mathcal{D}_f} \boldsymbol{f}_q \langle f_q | \gamma \rangle \; ; \quad \langle f_q | \gamma \rangle = \sum_{l=1}^{w} \langle f_q | e_l \rangle \langle e_l | \gamma \rangle, \tag{6}$$

where $\langle f_q | e_l \rangle = \frac{1}{\sqrt{w}} \mathrm{e}^{-\mathrm{i}f_q(l-l_0)}$, and we used the bracket notation to represent the inner product between vectors ($\langle e_l | \gamma \rangle \equiv \boldsymbol{e}_l^{\mathrm{T}} \boldsymbol{\gamma}$, etc). We call the representation based on $\{\boldsymbol{f}_q\}$ the Fourier representation (FR). If $\boldsymbol{\gamma}$ is the expression of real-valued time series data, the weight on l must be real, so it follows that

$$\langle e_l | \gamma \rangle = \frac{1}{\sqrt{w}} \sum_{q \in \mathcal{D}_f} |\langle f_q | \gamma \rangle| \cos(f_q l + \phi), \tag{7}$$

where $\phi = -f_q l_0 + \arg\langle f_q | \gamma \rangle$.

4 Density Matrix Formulation of k-Means

4.1 Objective Function of k-Means

Consider a general k-means clustering task for a set of vectors $\{\boldsymbol{s}_p \in \mathcal{H} \mid p = 1, 2, ..., n\}$. It is well-known that the k-means algorithm attempts to minimize the sum-of-squared (SOS) error [4]:

$$E = \sum_{j=1}^{k} \sum_{p \in \mathcal{C}_j} \left|\left| \boldsymbol{s}_p - \boldsymbol{m}^{(j)} \right|\right|^2 = \sum_{p=1}^{n} \langle s_p | s_p \rangle - \sum_{j=1}^{k} |\mathcal{C}_j| \langle m^{(j)} | m^{(j)} \rangle, \tag{8}$$

where $\mathcal{C}_j$ and $|\mathcal{C}_j|$ represent the members of the j-th cluster and the number of members in the cluster, respectively. The centroid of $\mathcal{C}_j$ is denoted by $\boldsymbol{m}^{(j)}$.

The first term does not depend on the clustering. For the second term, E_2, by substituting the definition of the centroid $\boldsymbol{m}^{(j)} = \frac{1}{|\mathcal{C}_j|} \sum_{p \in \mathcal{C}_j} \boldsymbol{s}_p$, it becomes

$$E_2 = -\sum_{j=1}^{k} \frac{1}{|\mathcal{C}_j|} \sum_{p,r \in \mathcal{C}_j} \langle s_p | s_r \rangle. \tag{9}$$

To remove the restricted summation, we introduce an indicator vector $\boldsymbol{u}^{(j)} \in \mathcal{H}$, where $\langle s_p | u^{(j)} \rangle = 1/\sqrt{|\mathcal{C}_j|}$ for $\boldsymbol{s}_p \in \mathcal{C}_j$ and 0 otherwise, to have

$$E_2 = -\sum_{j=1}^{k} \sum_{p,r=1}^{n} \langle u^{(j)} | s_p \rangle \langle s_p | s_r \rangle \langle s_r | u^{(j)} \rangle.$$

Now we introduce a linear operator $\hat{\rho}$ as

$$\hat{\rho} = \sum_{p=1}^{n} \boldsymbol{s}_p \boldsymbol{s}_p^{\mathrm{T}}$$

and call $\hat{\rho}$ the *density matrix*, following the statistical-mechanical terminology. Since the $\boldsymbol{s}_p$s are generated by the SW technique, we see

$$\hat{\rho} \doteq \sum_{l=1}^{n} \hat{\tau}(l)^{\mathrm{T}} \boldsymbol{\Gamma} \boldsymbol{\Gamma}^{\mathrm{T}} \hat{\tau}(l) \tag{10}$$

holds, where "$\doteq$" means "the left and the right sides have the same matrix elements when represented in $\mathcal{H}$ (not $\mathcal{H}_0$)".

Using $\hat{\rho}$, we get the final form of the objective function as

$$E_2 = -\sum_{j=1}^{k} \langle u^{(j)} | \hat{\rho}^2 | u^{(j)} \rangle, \tag{11}$$

where $\langle \cdot | \hat{o} | \cdot \rangle$ is defined as $\langle \cdot | \hat{o} \cdot \rangle$ for $\forall \hat{o} \in \mathcal{L}$. The k-means clustering task has now been reduced to seeking the solution $\{\boldsymbol{u}^{(j)}\}$ which minimizes E_2.

4.2 Connection to Eigen Problem

To this point, the vector $\boldsymbol{u}^{(j)}$ has been an artificially defined indicator to simplify the objective in Eq. (9). From the original definition, it is easy to see that $\{\boldsymbol{u}^{(j)}\}$ satisfy

$$\sum_{p=1}^{n} \langle u^{(i)}|s_p\rangle\langle s_p|u^{(j)}\rangle = \langle u^{(i)}|\hat{\rho}|u^{(j)}\rangle = \delta_{i,j}. \tag{12}$$

Now we relax the original binary restriction, and take this as the new restriction on the optimization problem, so that the k-means task is reduced to the generalized eigen problem which minimizes E_2 subject to Eq.(12). This eigen problem can be written as

$$\hat{\rho}\boldsymbol{u}^{(j)} = \lambda_j \boldsymbol{u}^{(j)} \ \text{ s.t. } \ \langle u^{(i)}|u^{(j)}\rangle = \delta_{i,j}, \tag{13}$$

where λ_j is the eigenvalue corresponding to the eigenstate $\boldsymbol{u}^{(j)}$ labeled in descending order of the eigenvalues. In the SR, $\langle e_l|\hat{\rho}|e_{l'}\rangle$ corresponds to the (l, l') element of $\mathsf{H}\mathsf{H}^{\mathrm{T}}$, where $\mathsf{H} = [\boldsymbol{s}_1, ..., \boldsymbol{s}_n]$ (note that H has n columns by PBC). Thus, Eq. (13) can be written as

$$\mathsf{H}\mathsf{H}^{\mathrm{T}}\boldsymbol{u}^{(j)} = \lambda \boldsymbol{u}^{(j)}.$$

This equation also shows the $\boldsymbol{u}^{(j)}$s are the left singular vectors of H.

4.3 Cluster Centers and Eigenstates

Apart from the formal definition as the (relaxed) indicator, let us further consider the meaning of $\boldsymbol{u}^{(j)}$. Before the relaxation, the indicator satisfied

$$\boldsymbol{m}^{(j)} \equiv \frac{1}{|\mathcal{C}_j|} \sum_{p \in \mathcal{C}_j} \boldsymbol{s}_p = \frac{1}{\sqrt{|\mathcal{C}_j|}} \sum_{p=1}^{n} \boldsymbol{s}_p \langle s_p|u^{(j)}\rangle.$$

After the relaxation, $\boldsymbol{u}^{(j)}$ is the eigenstate of $\hat{\rho} = \sum_p \boldsymbol{s}_p \boldsymbol{s}_p^{\mathrm{T}}$. Thus, it follows that *the k-means cluster centers correspond to the eigenstates of $\hat{\rho}$*, or

$$\boldsymbol{m}^{(j)} \propto \boldsymbol{u}^{(j)}. \tag{14}$$

Note that our formulation directly seeks the cluster centers as the eigen vectors. This is in contrast to the standard spectral formulations [3].

Now, we summarize this section as Theorems:

Theorem 1. *The eigenstates of $\hat{\rho}$, which can be computed also as the left singular vectors of H, minimize the SOS objective.*

Theorem 2. *The eigenstates of $\hat{\rho}$ formally correspond to the k-means cluster centers.*

In spite of this, the correspondence between the k-means and our spectral formulation is not perfect. The major discrepancy comes from the fact that the eigenstates must be orthogonal to each other. The problem is that the cluster centers are not necessarily orthogonal in general. One reasonable expectation is that the top eigenstate $\boldsymbol{u}^{(1)}$ would be a good estimator representing the averaged direction of a few of the major k-means clusters. For the other eigenstates, the direction would be more or less influenced by the top one. [2] We will discuss this topic theoretically and experimentally later.

5 Fourier Representation of $\hat{\rho}$

5.1 The $w = n$ Case

Let us consider the extreme case of $w = n$. In this case, $\mathcal{H}$ ($= \mathcal{H}_0$) can be thought of as periodic, so that the Fourier state $\boldsymbol{f}_q$ is the exact eigenstate of $\hat{\tau}(l)$. Explicitly,

$$\hat{\tau}(l)\boldsymbol{f}_q = \frac{1}{\sqrt{w}} \sum_{l'=1}^{n} \mathrm{e}^{\mathrm{i}f_q(l'-l_0)} \boldsymbol{e}_{l'+l} = \mathrm{e}^{-\mathrm{i}f_q l} \boldsymbol{f}_q. \tag{15}$$

Here we used the fact that $\mathrm{e}^{\mathrm{i}f_q n} = 1$ if $f_q = 2\pi q/n$.

Using Eqs. (10) and (15), we can calculate $\langle f_q|\hat{\rho}|f_{q'}\rangle$ as

$$\sum_{l=1}^{n} \langle f_q|\hat{\tau}(l)^{\mathrm{T}}|\Gamma\rangle\langle\Gamma|\hat{\tau}(l)|f_{q'}\rangle = \sum_{l=1}^{n} \langle f_q|\Gamma\rangle\langle\Gamma|f_q\rangle \mathrm{e}^{\mathrm{i}(f_q \quad f_{q'})l} = n|\langle f_q|\Gamma\rangle|^2\delta_{q,q'}, \tag{16}$$

which means the matrix representation of $\hat{\rho}$ is diagonal in FR. Thus, we conclude that *the Fourier state itself is the eigenstate of $\hat{\rho}$ completely independently of the input data*. Which f_q is chosen depends on the magnitude of $|\langle f_q|\Gamma\rangle|^2$, the *power* of the Fourier component. Note that the eigenstate must be a pure sinusoid even when the power spectrum does not have any dominant f_q[3]. When a q_1 was chosen, the resultant distribution is sinusoidal with the wave number f_{q_1} (see Eq. (7)). Thus, based on Theorems 1 and 2, the k-means cluster centers are expected to be approximated by the sinusoids apart from the orthogonality problem.

5.2 The $w < n$ Case

For $w < n$, the $\boldsymbol{f}_q$s are not exactly the eigenstates of $\hat{\tau}(l)$, since $\mathcal{H}$ cannot be thought of as periodic. As a result, we have the matrix elements like

$$\langle f_q|\hat{\rho}|f_{q'}\rangle \approx n|\langle f_q|\Gamma\rangle|^2\delta_{q,q'} + \sum_{l=1}^{n} \mathrm{e}^{\mathrm{i}l\Delta_{q'q}} J_l(q,q'), \tag{17}$$

[2] The $k = 1$ case is special. The cluster center is the simple mean vector, and can be written as $|m\rangle = \sqrt{w}\bar{x}|f_0\rangle$, where $\bar{x}$ denotes the mean of Γ over the whole domain. This gives a constant distribution, having no relationship with the $\boldsymbol{u}^{(j)}$s.

[3] This is not the case when some of the $|f_q|$s have exactly the same power, but it is unlikely in real-world time-series data under normal conditions.

instead of Eq. (16). It is straightforward to get the exact expression of $J_l(q, q')$ although we do not show it here. However, under normal conditions, we can assume that the first term is the leading term since $n \gg 1$ and phase cancellations are unavoidable in the second term. In particular, if the power spectrum has a single-peaked structure at $|f_q|$, which is the case in the CBF data (see the next section), the top eigenstate will be well approximated by the $\boldsymbol{f}_{|q|}$, irrespective of the details of the spectrum. As a result, Eq. (7) reads

$$\langle e_l | u \rangle \propto \cos(f_q l + \phi). \tag{18}$$

Since l_0 was arbitrary, the real number ϕ is also arbitrary. From this, we can naturally understand the unpredictability of the additive phase as stated in Observation 1. Now we get Theorem 3, which directly explains Observation 2:

Theorem 3. *When a $|q|$ is dominant, the singular vectors of H are well approximated by sinusoids with the wavelength of $w/|q|$, irrespective of the details of the input time series data.*

In addition, by considering Theorems 1 and 2, the k-means cluster centers will be sinusoidal except for the orthogonality problem. This is the mathematical explanation of Observation 1.

When the power spectrum is almost flat, the eigenvectors will be mixtures of many f_qs, so that the cluster centers will be far from pure sinusoids.

5.3 Optimizing the Relative Phases

If the data has a dominant q, the subsequences can be approximated as

$$\boldsymbol{s}_p = \sum_{q' \in \mathcal{D}_f} \boldsymbol{f}_{q'} \langle f_{q'} | s_p \rangle \approx \sum_{q' \in \mathcal{D}_f} \mathrm{e}^{\mathrm{i} f_{q'} p} \boldsymbol{f}_{q'} \langle f_{q'} | \Gamma \rangle \approx \boldsymbol{f}_q \langle f_q | \Gamma \rangle \mathrm{e}^{\mathrm{i} f_q p}. \tag{19}$$

Define $\boldsymbol{g}_{q,\phi} \in \mathcal{H}$ by $\langle e_l | g_{q,\phi} \rangle = \cos(f_q l + \phi)$. Equation (19) means that the k-means STSC is reduced to that for $\{\boldsymbol{g}_{q,\phi}\}$ with uniformly distributed ϕ.

Since $\{\boldsymbol{g}_{q,\phi}\}$ consists of sinusoids of f_q, the cluster centers must be sinusoids of f_q. Let the cluster centers be $\boldsymbol{g}_{q,\phi_j}$ $(j = 1, ..., k)$. The distribution of ϕ may be modeled as a continuous uniform distribution over $[0, 2\pi)$. The SOS objective is now written as

$$E(\phi_1, ..., \phi_k) = \frac{1}{2\pi} \int_0^{2\pi} \mathrm{d}\phi \sum_{j=1}^{k} \theta_j(\phi) e(\phi, \phi_j), \tag{20}$$

where $1/(2\pi)$ represents the probability density of ϕ, and $e(\phi, \phi_j) \equiv ||\boldsymbol{g}_{q,\phi} - \boldsymbol{g}_{q,\phi_j}||^2 = 4 \sin^2 \frac{\phi - \phi_j}{2}$. The function $\theta_j(\phi)$ indicates cluster assignment, and takes the value 1 when the j-th cluster center is closest to ϕ, or 0 otherwise. For example, if we have $\phi_1 < \phi_2 < \phi_3$ when $k = 3$, $\theta_2(\phi)$ will be 1 for $\frac{\phi_1 + \phi_2}{2} \leq \phi < \frac{\phi_2 + \phi_3}{2}$ and 0 otherwise. Solving the minimization problem of E w.r.t. the phases is straightforward but tedious. However, it is intuitively clear that the

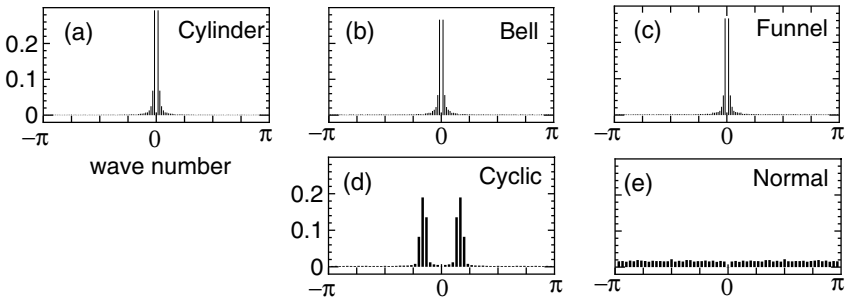


Fig. 3. Power spectra of each instance of the data

most balanced assignment is optimal. In fact, Eq. (20) is symmetric w.r.t. j, so the solution should be also symmetric if it is the unique solution. Now we arrive at Theorem 4, which summarizes the theoretical proof of the phase question in Observation 1:

Theorem 4. *If a Γ has a dominant f_q, the k-means STSC is reduced to that for uniformly distributed sinusoids of f_q. The optimal cluster centers are separated by a phase of integral multiples of $2\pi/k$.*

6 Experiments

6.1 Cylinder-Bell-Funnel Data

The CBF data [7] includes three types of patterns literally having Cylinder, Bell, and Funnel shapes. We randomly generated 30 instances for each type (examples in Fig. 1 (b)) with a fixed length of 128 ($= w$) using Matlab code provided by [7]. We also concatenated them in order after standardizing each one (zero mean and unit variance). We did 100 random restarts and chose the best one in the k-means calculation.

Figures 3 (a)-(c) show the power spectra of each instance as a function of the wave number. To handle the variation of the instances, we simply averaged the resultant spectra for all instances. We see that the most of the weight is concentrated on the $|q| = 1$ component in all of the cases. The f_0 component is naturally missing because of the standardization.

The results of k-means and SVD were shown in Fig. 2. The wavelength of w can be understood from the large $|q| = 1$ weight in Fig. 3 (a)-(c). Due to the orthogonality condition, the third singular vector necessarily has a wavelength of about $w/2$. This is an example of the difference between the two formulations in how the calculated cluster centers interact with each other. Apart from this, our formulation is completely consistent with the results.

6.2 Synthetic Control Chart Data

The Synthetic Control Chart (SCC) data [7] consists of six types of 100 instances, each with 60 data values. Out of the six types, we focus on the Cyclic and

Normal types (Fig. 4), which have very different (averaged) power spectra from the CBF spectra, as shown in Fig. 3. We see that the weight is concentrated on the wavelengths of $\frac{w}{4}$, $\frac{w}{5}$, and $\frac{w}{6}$ in the Cyclic data ($w = 60$). In contrast, the distribution is almost flat for the Normal data, as expected for white noise.

We made a concatenated data set with 100 standardized Normal instances followed by 100 standardized Cyclic instances. Figures 5 (a) and (b) show the k-means cluster centers ($k = 2$, the best one among 100 random restarts) and the two highest singular vectors, respectively. We set $w = 60$. Since the $\boldsymbol{s}_p$s in the Normal part do not favor any particular direction, the clustering results seem to be dominated by the Cyclic part. In both figures, amplitude-modulated sinusoids with a periodicity of about $w/5$ are observed instead of pure sinusoids. The waves are separated by the phase intervals which can be naturally understood from Theorem 4 for (a) and from the orthogonality condition for (b).

The amplitude modulation can be understood as *beat* in physics. As shown in Fig. 3, the Cylinder part is dominated by the $f_{|4|}, f_{|5|}$ and $f_{|6|}$ components. Since SVD extracts the major direction of $\{\boldsymbol{s}_p\}$, the top singular vector $\boldsymbol{u}$ will be approximated as a linear combination of those components like

$$\langle e_l|u\rangle \approx \sum_{q=4}^{6} c_q \cos\left[f_q(l-l_0)\right].$$

Within the accuracy up to the order of $(2\pi/w)^2$, it reads

$$\langle e_l|u\rangle \propto \mathrm{e}^{-\frac{1}{2}\Delta_2{}^2(l-l_0)^2} \cos\left[(f_q-\Delta_1)(l-l_0)\right], \tag{21}$$

where $\Delta \equiv 2\pi/w$, and

$$\frac{\Delta_1}{\Delta} = \frac{c_4 - c_6}{c_4 + c_5 + c_6}\ , \quad \frac{\Delta_2}{\Delta} = \frac{\sqrt{4c_4c_6 + c_5(c_6+c_4)}}{c_4+c_5+c_6}.$$

To derive this, we used Taylor expansion formulas with $\epsilon \ll 1$ such as

$$\ln\left(c_5 + c_6\mathrm{e}^{\mathrm{i}\epsilon}\right) \approx \ln(c_5+c_6) + \frac{\mathrm{i}\epsilon c_6}{c_5+c_6} - \frac{\epsilon^2}{2}\frac{c_5c_6}{(c_5+c_6)^2}.$$

The line shape in Eq. (21) is known as beat in physics. If we set $c_q \propto |\langle f_q|\Gamma\rangle|$ (in the Cyclic part), we get $\Delta_1 = 0.1\Delta$ and $\Delta_2 = \Delta/1.3$ from Fig. 3 (d). This leads to a sine wave with wavelength $w/4.9$ modulated by a beat wavelength of $1.3w$. Except for the region where $\Delta(l-l_0) \simeq 1$, Equation (21) explains Fig. 5 reasonably well.

It is interesting to see what happens when we have only the Normal data. As expected, the resulting cluster centers are far from sinusoids when $w = 60$ (not shown). However, STSC produces sinusoids when $w = n$ (=6000), despite the white noise nature of the data. Our theory clearly explains this counter intuitive result. As discussed in Subsection 5.1, the top singular vector must be the pure sinusoid of the largest power. In this case, we have the largest power at $|f_q| = 0.358$ in Fig. 6 (a) (marked by the triangles). Thus, the wavelength must

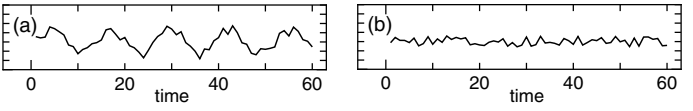


Fig. 4. Examples of (a) Cyclic and (b) Normal instances in the SCC data

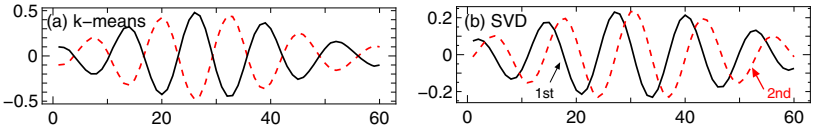


Fig. 5. (a) The k-means cluster centers and (b) the first and second singular vectors of H for the concatenated Normal-Cyclic data

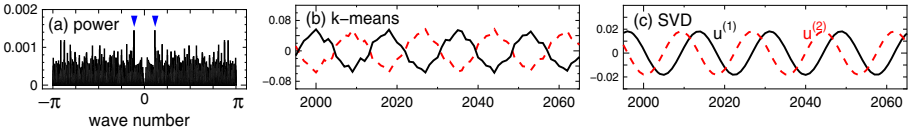


Fig. 6. The results for the concatenated Normal data with $w = n = 6000$. (a) The power spectrum, (b) a segment of the k-means cluster centers ($k = 2$), and (c) a segment of the top two left singular vectors.

be $2\pi/|f_q| = 17.6$, which is completely consistent with Fig. 6 (c). In addition, we see that the singular vector is a good estimator of the k-means cluster center by comparing Figs. 6 (b) and (c).

While some authors attribute the sinusoidal patterns to simple smoothing effects caused by superposition of slightly shifted subsequences [1], such a discussion clearly fails to explain the origin of the sinusoidal curves for the Normal data. Also, it is not capable of explaining the beat waves.

7 Concluding Remarks

We have performed a theoretical analysis of the sinusoid effect in STSC. In particular, we pointed out that the k-means clustering task can be reduced to the eigen problem of the density matrix $\hat{\rho}$. Thanks to the translational symmetry of $\hat{\rho}$, the eigenstate can be approximated by a Fourier state if a single $|f_q|$ forms a conspicuous single peak in DFT. We also found that the k-means cluster centers produce beat waves (Fig. 5) when a few neighboring frequencies are dominant.

Mathematically, the sinusoid effect can be understood from the fact that the Fourier states are the irreducible representations of the translational group. In another paper [5], we used a point group for pattern discovery. This paper also

can be seen as one of the first studies which introduce the concept of group into machine learning.

Our theory also provides a practical basis for attempts to make STSC meaningful. As long as the coherent superposition is used to define the cluster centers, sinusoid pseudo-patterns are more or less unavoidable. One possibility is to utilize incoherent superposition of subsequences. Medoid-based methods are perhaps the simplest way to use the incoherence, and are known to give better results than the simple STSC. Detailed discussion on this issue will appear elsewhere.

Acknowledgement

The author thanks S. Jacobs for carefully reading the manuscript.

References

1. J. Chen. Making subsequence time series clustering meaningful. In *Proc. IEEE Intl. Conf. Data Mining*, pages 114–121, 2005.
2. I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means, spectral clustering and normalized cuts. In *Proc. Tenth ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, pages 551–556, 2004.
3. C. Ding and X. He. K-means clustering via principal component analysis. In *Proc. Intl. Conf. Machine Learning*, pages 225–232, 2004.
4. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed.)*. Wiley Interscience, 2000.
5. T. Idé. Pairwise symmetry decomposition method for generalized covariance analysis. In *Proc. IEEE Intl. Conf. Data Mining*, pages 657–660, 2005.
6. E. Keogh. Data mining and machine learning in time series databases. In *Tutorial Notes of the Tenth ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*. 2004.
7. E. Keogh and T. Folias. The UCR time series data mining archive [`http://www.cs.ucr.edu/ eamonn/TSDMA/index.html`]. 2002.
8. E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *Proc. IEEE Intl. Conf. Data Mining*, pages 115–122. IEEE, 2003.
9. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems, 14*, pages 849–856, 2001.
10. H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems, 14*, pages 1057–1064, 2001.

Transductive Learning for Text Classification Using Explicit Knowledge Models

Georgiana Ifrim and Gerhard Weikum

Max-Planck Institute for Informatics,
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
{ifrim, weikum}@mpi-inf.mpg.de

Abstract. We present a generative model based approach for transductive learning for text classification. Our approach combines three methodological ingredients: learning from background corpora, latent variable models for decomposing the topic-word space into topic-concept and concept-word spaces, and explicit knowledge models (light-weight ontologies, thesauri, e.g. WordNet) with named concepts for populating latent variables. The combination has synergies that can boost the combined performance. This paper presents the theoretical model and extensive experimental results on three data collections. Our experiments show improved classification results over state-of-the-art classification techniques such as the Spectral Graph Transducer and Transductive Support Vector Machines, particularly for the case of sparse training.

Keywords: transductive learning, latent models, expectation-maximization.

1 Introduction

1.1 Motivation

Many applications require classification models able to learn from few labeled data and rich background corpora. Examples could be Amazon organizing book descriptions into pre-defined categories, Google or Yahoo! classifying crawled Web pages into topic directories for more convenient access, or Wikipedia categorizing encyclopedia articles for better search and browsing. A learning paradigm in which the data collection to be automatically labeled is available beforehand is referred to as *transductive inference*.

Transductive learning is particularly attractive for text classification with very few explicitly labeled training documents, which happens whenever human assessment is the (time or cost) bottleneck on rapidly growing and highly diverse corpora. In such a setting, being able to harness the feature distributions and relations among the unlabeled documents is an important asset to improve the classifier. Another potentially beneficial asset is explicit knowledge about concepts, words and phrases that express concepts, and the semantic relations among concepts (e.g., hyponymy). Such knowledge sources may be given in the form of an ontology or thesaurus.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 223–234, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Prior work has mostly pursued "latent semantic" models such as spectral analysis, and the few approaches that have attempted to leverage explicit knowledge sources focused on concept-aware feature spaces and did not integrate concept-word relationships into the learning procedure itself. Moreover, many of these prior methods faced difficult model selection problems regarding feature engineering and parameter tuning. In the current paper, we develop a novel approach, based on a generative model with explicit concepts, that combines background ontologies with transductive learning on large corpora. Our approach aims to make classifiers more robust and improve classification accuracy with very few training data and as little model tuning as possible.

1.2 Contribution

Our approach is based on a generative model for text documents where words are generated by concepts which in turn are generated by topics. We postulate conditional independence between words and topics given the concepts. Once the corresponding probabilities for word-concept and concept-topic pairs are estimated, we can use Bayesian inference to compute the probability that a test document with known words but unobservable concepts belongs to a certain topic. The concepts are used as latent variables here, but unlike earlier work on spectral decomposition and latent semantic models [2], [6], [8] our concepts are named and can be explicitly identified in the underlying ontology or thesaurus. We employ an iterative EM (expectation-maximization) procedure for maximum-likelihood parameter estimation. The effectiveness of EM greatly benefits from a judicious initialization step that estimates word-concept probabilities on the *unlabeled* part of the document collection, thus leading us to a transductive learning method.

For illustration, consider a training document that contains the ambiguous word *Java* and belongs to the topic *geography*. Our method uses the background ontology and the word-occurrence contexts of the entire corpus, comprising both training and unlabeled documents, to map the word *Java* to its corresponding concept *Java, island belonging to Indonesia* and, at the same time, estimate the probability distribution of words that can express this concept (with, e.g., "Sumatra" or "Jakarta" having higher probabilities than "method" or "inheritance"). The statistics gained from the full corpus are fed into the EM procedure to further improve the concept-word estimates and to learn the concept-topic distribution. This technique populates only latent concept variables that have significant probability of being associated with the words occurring in documents, thus making the EM iterations much more efficient and effective. We consider the following as our main contributions:

1. By using explicit concepts from an ontology or thesaurus and by using a heuristic technique for bootstrapping the word-to-concept mapping, we avoid the model selection problem inevitably faced by all techniques based on latent dimensions (i.e., choosing an appropriate number of dimensions).

2. By the same token, we avoid the combinatorial explosion in the space of parameters to be estimated (i.e., concept-word pairs), and we largely eliminate the need for parameter smoothing (which is often a very tricky issue).
3. The initial word-to-concept mapping is very beneficial for fast convergence of EM, and reduces the danger of getting stuck in local maxima of the likelihood function. The latter is an issue especially with very few training data.
4. Our method provides an intuitive and effective way of exploiting the available resources, unlabeled documents from the full corpus and ontological relationships among concepts, resulting in improved classification accuracy with few training data.
5. Our approach is more robust in the sense that it requires considerably less tuning than other transductive methods.

In our experiments, with real-life datasets from the Reuters newswire corpus, Amazon book reviews, and Wikipedia articles, we compare our method with the Spectral Graph Transducer [11] and Transductive SVM classifiers [12] and demonstrate the viability and superiority of our approach.

1.3 Related Work

There are many approaches to transductive learning. Transductive SVMs were introduced by [20] and applied to text classification by [1], [12]. They exploit the structure in both training and test data for better positioning the maximum margin hyperplane. However, as shown empirically and theoretically in [21], learning a maximum margin from the unlabeled data in order to assign the labels is unreliable. Generative model based approaches can exploit the information in the unlabeled test collection for better estimating the generating distribution. An example is the usage of unlabeled data for re-weighting labeled examples [16]. Other methods represent the dataset as a graph and exploit the structure in the entire dataset in search for mincuts [4] or for min average cuts [11].

Various latent aspect models have been proposed in the literature [2], [6], [8]. In all this prior work, latent dimensions are mathematical constructions without any association to explicitly denoted concepts. This can make the interpretation of classification results by such methods difficult or unintuitive. Moreover, all prior methods require careful tuning of the number of latent dimensions. Finally, latent aspect models are either completely unsupervised or used as preprocessing step for feature engineering.

Explicit knowledge sources like ontologies, thesauri, or dictionaries have been used in prior work on text classification only for feature engineering, e.g., constructing composite words or phrases as features based on a thesaurus. Most notably, the WordNet thesaurus [7] has been leveraged in various approaches of this kind [3], [19]. In contrast, we propose integrating explicit knowledge about word-concept relationships into the learning procedure itself.

2 Transductive Latent Model

In this section we introduce our framework and the theoretical model proposed.

2.1 Generative Model

We are given a large *document collection*, $D = \{d_1, \ldots, d_r\}$, split into a small *training* set with known *topic labels*, $T = \{t_1, \ldots, t_m\}$, and a large test set with unknown topic labels. We have access to an *ontology graph of concepts*, $C = \{c_1, \ldots, c_k\}$, where each concept has a set of synonyms and a short textual description, and is related to other concepts by semantic edges (e.g., hypernymy/hyponymy relations). From the given collection we can select a set of *features*, $F = \{f_1, \ldots, f_n\}$ (*words or phrases*). A document is considered to be a multiset of features. Our generative model for feature-topic co-occurrence can be described as:

1. Select a topic t with probability $P[t]$;
2. Pick a latent variable c with probability $P[c|t]$, the probability that concept c describes topic t;
3. Generate a feature f with probability $P[f|c]$, the probability that feature f means concept c.

The pairs (f, t) can be directly observed, while the existence of concepts implies some form of word sense disambiguation; concepts are treated as latent variables.

Mathematically, our model is similar to the *aspect model* developed in [8]. However, while in the aspect model the number of concepts has to be known a priori and the concepts themselves are implicit (an abstract mathematical notion), our model uses existing knowledge resources (ontologies, thesauri) to identify and select explicit concepts at runtime. A similar model was used in our earlier work [9] for the inductive learning setting.

2.2 Learning Model Parameters

For tractability reasons, the generative model is based on two independence assumptions: The observation pairs (f, t) are generated independently and the features f are conditionally independent of the topics t, given the latent variable c: $P[(f, t)|c] = P[f|c] \cdot P[t|c]$. Using explicit concepts from a fine-grained ontology like WordNet helps alleviating the impact of these assumptions on the model robustness. To describe the generative process of an observation (f, t), we sum up over all the possible values that the latent variables might take

$$P[f, t] = \sum_{c \in C} P[c] \cdot P[(f, t)|c]. \tag{1}$$

We assume that the (f,t) pairs are generated from a multinomial distribution. Thus, the likelihood of the observed (f, t) samples can be expressed as:

$$\begin{aligned} L &= \Pi_{f \in F, t \in T} P[f, t]^{n(f,t)} \\ &= \Pi_{f \in F, t \in T} (\sum_{c \in C} P[c] \cdot P[(f, t)|c])^{n(f,t)} \\ &= \Pi_{f \in F, t \in T} (\sum_{c \in C} P[f|c] \cdot P[c|t] \cdot P[t])^{n(f,t)} \end{aligned} \tag{2}$$

where $n(f,t)$ is the number of occurrences of feature f in the training set of topic t (multiple occurrences in one document count multiple). The learning problem consists in estimating the parameters ($P[f|c]$, $P[c|t]$, $P[t]$) of the generating distribution, given the observed (f,t) samples. This can be formally expressed as a maximization of the observed data log-likelihood:

$$l = \sum_{f \in F, t \in T} n(f,t) \cdot log(\sum_{c \in C} P[f|c] \cdot P[c|t] \cdot P[t]) \tag{3}$$

Due to the sum inside the logarithm direct maximization of the log-likelihood by partial derivatives is difficult. To overcome this problem, we employ an Expectation-Maximization (EM) algorithm (see [10] for more details). The EM algorithm works by 2 iterative steps:

- **E-Step:** Expectation step, in which posterior probabilities are estimated for the latent variables, taking as evidence the observed data. For calculating the estimates of the E-step, we use Bayes' formula:

$$P[c|(f,t)] = \frac{P[(f,t)|c] \cdot P[c]}{\sum_{c' \in C} P[(f,t)|c'] \cdot P[c']} = \frac{P[f|c] \cdot P[c|t] \cdot P[t]}{\sum_{c' \in C} P[f|c'] \cdot P[c'|t] \cdot P[t]} \tag{4}$$
$$= \frac{P[f|c] \cdot P[c|t]}{\sum_{c' \in C} P[f|c'] \cdot P[c'|t]}$$

- **M-Step:** Maximization step, in which the current parameters are updated based on the expected complete data log-likelihood, which depends on the posterior probabilities estimated in the E-Step. Detailed justification for the equations (5), (6) and (7) is given in [10].

$$P[f|c] = \frac{\sum_{t \in T} n(f,t) P[c|(f,t)]}{\sum_{f' \in F} \sum_{t \in T} n(f',t) P[c|(f',t)]} \tag{5}$$

$$P[c|t] = \frac{\sum_{f \in F} n(f,t) P[c|(f,t)]}{\sum_{c' \in C} \sum_{f \in F} n(f,t) P[c'|(f,t)]} \tag{6}$$

$$P[t] = \frac{\sum_{f \in F, c \in C} n(f,t) P[c|(f,t)]}{\sum_{t' \in T} \sum_{f \in F, c \in C} n(f,t') P[c|(f,t')]} \tag{7}$$

In order to compute the conditional probability of a document given a topic $P[d|t]$, postulating feature independence, the estimates $P[f|c]$ and $P[c|t]$ are used:

$$P[d|t] = \Pi_{f \in d} P[f|t] = \Pi_{f \in d} \sum_{c \in C} P[f|c] \cdot P[c|t] \tag{8}$$

Once we have estimates for the marginal distribution describing the generative model, we can use Bayes' rule to reverse the model and predict which topic generated a certain document:

$$P[t|d] = \frac{P[d|t] \cdot P[t]}{P[d]} = \frac{P[d|t] \cdot P[t]}{\sum_{t' \in T} P[d|t'] \cdot P[t']} \tag{9}$$

We then substitute (8) into (9) and have a decision procedure for the classifier.

2.3 Problems and Solutions

EM faces two major problems:

1. The combinatorial explosion of the variable space in the model, since the number of parameters is directly proportional to the cross-product of the number of features, concepts and topics. These parameters are sparsely represented in the observed training data.
2. The possibility of slow convergence or convergence to a local maximum of the likelihood function and missing the global optimum.

For the first problem, it is desirable to **prune the parameter space** to reflect only the meaningful latent variables. We propose two techniques to this end.

Feature Selection. From the given collection we extract a set of features using $tf \cdot idf$ ranking as a quality measure. For each term we compute its frequency in the entire collection (tf) and its inverse document frequency (idf). We normalize these quantities [5] and rank the terms according to their $tf \cdot idf$ value. As a preprocessing step, we extract compound words, e.g. "exchange market", using a sliding window parser and a background dictionary (e.g WordNet), and treat them as individual features. This step can play a role in capturing the semantics of interesting and common language constructions; it also reduces some of the computational overhead, while helping model robustness: many compounds have only one meaning, e.g. "exchange market", versus "exchange" and "market".

Concept Set Selection. We use the WordNet thesaurus [7] as the basis for our ontology graph. WordNet contains around 150,000 concepts (word senses) linked by hierarchical relations. Using the entire set of concepts would result in a high computational overhead and a high amount of noise. A better approach is to select from the ontology only a subset of concepts that reflects the semantics of the data collection. We call this the *candidate set of concepts*. This set is selected in a preprocessing step, before running the EM algorithm; details are given below. The size of this subset is only a few thousands concepts, as opposed to some hundred-thousands available in the ontology.

For the second problem, slow or suboptimal convergence, it is desirable to **pre-initialize the model parameters** to values that are close to the global maximum of the likelihood function. In order to get a good initialization for our parameters $P[f|c]$ and $P[c|t]$ we use a similarity-based mapping approach. The technique consists of mapping features to concepts and concepts to topics, based on *similarity measures* between contexts. The WordNet thesaurus can be seen as a directed acyclic graph (DAG) where the nodes are the different concepts and the edges are semantic relationships [7]. Let w be a word that we want to map to the ontological senses. First, we query WordNet for the possible meanings of word w. Let $\{c_1, \ldots, c_m\}$ be the set of meanings associated with w. By taking also the synonyms of these word senses, we can form *synsets* for each of the word meanings. Next, we apply a word sense disambiguation step by computing the overlap between the local contexts of both the word observed in a document and each of

its possible meanings. This type of approach is commonly used in the word sense disambiguation literature. For each occurrence of word w in the text collection, its local context is a text window around its offset; the context for the concept is taken from the ontology: for each sense c_i we take its hypernyms, hyponyms, holonyms and their short textual descriptions. The context of a concept in the ontology graph can be taken until a certain depth, depending on the amount of noise one is willing to introduce in the disambiguation process. In this work we use depth 2. For each of the candidate senses c_i, we compute the cosine similarity between the tf vectors of $context(w)$ and $context(c_i)$, $i \in \{1, \ldots, m\}$. Only the most dominant meaning (over all occurrences of w) is kept in the concept space. For words having multiple meanings, our hypothesis is that "secondary" meanings are introduced by their corresponding features, e.g., for the word *Java* having meanings *island* and *coffee*, if *island* is selected as the main meaning of *Java*, the second meaning *coffee* can still be introduced in the concept space by occurrences of words like *espresso, latte, coffee beans*, etc.

In a similar fashion, we relate concepts to topics based on similarity of contexts. The context for a topic t is defined to be the bag-of-features selected from the training collection by decreasing Mutual Information (MI) [14] value. For our implementation, we used the top 50 terms with regard to MI rank. Once we have computed all the similarities for the (feature, concept) and (concept, topic) pairs, we normalize them, and interpret them as estimates of the probabilities $P[f|c]$ and $P[c|t]$. In the $sim(f, c)$ and $sim(c, t)$ computations, we only consider the (f, c) and (c, t) pairs in the pruned parameter space. The computed values are then used for enhancing the EM algorithm in the model fitting process.

Transductive Learning. Our *feature-concept* mapping does not require labeled documents, thus it can be computed on the entire unlabeled collection. This has the effect of *drastically reducing the need for training data.* The *concept-topic* mapping might be poorly estimated from sparse training as the topics might be poorly described in the training set. This problem can be alleviated by extending the topic description using background corpora. For example, we can improve the training description of the topic *Biology* by using a better description from an encyclopedia, e.g. the Biology page from Wikipedia. We note that this methodology of using background knowledge makes our model robust to variations in vocabulary or data distribution, problems that can occur when directly adding background data to the training set.

2.4 Enhanced EM Algorithm

The mapping step presented in Section 2.3 can be seen as defining a prior probability distribution on the model parameters $P[f|c]$, $P[c|t]$. This can be exploited in a Maximum A Posteriori (MAP) estimation of parameters, rather than simple maximum likelihood estimation. We denote by $\theta = (\theta_{f|c}, \theta_{c|t}, \theta_t)$, $\theta_{f|c} = P[f|c]$, $\theta_{c|t} = P[c|t]$, $\theta_t = P[t]$ our model parameters. Let $\theta_{f|c} \sim Dirichlet(\alpha_f^c)$ and $\theta_{c|t} \sim Dirichlet(\beta_c^t)$, where $\alpha_f^c = sim(f, c)$ for each $f \in F$ and $c \in C$ and

$\beta_c^t = sim(c,t)$ for each $c \in C$ and $t \in T$. Let θ_t be uniformly distributed, with density $g(\theta_t) = \frac{1}{|T|}$. The corresponding densities for $\theta_{f|c}$ and $\theta_{c|t}$ are:

$$g(\theta_{f|c}) \sim \Pi_{f \in F}\ {\theta_{f|c}}^{\alpha_f^c},\ \theta_{f|c} \geq 0,\ \alpha_f^c \geq 0,\ \sum_{f \in F} \theta_{f|c} = 1,\ \forall c \in C \tag{10}$$

$$g(\theta_{c|t}) \sim \Pi_{c \in C}\ {\theta_{c|t}}^{\beta_c^t},\ \theta_{c|t} \geq 0,\ \beta_c^t \geq 0,\ \sum_{c \in C} \theta_{c|t} = 1,\ \forall t \in T \tag{11}$$

Because $\theta_{f|c}$, $\theta_{c|t}$ and θ_t are independent random variables, their joint density can be written as: $g(\theta) = g(\theta_{f|c}, \theta_{c|t}, \theta_t) = g(\theta_{f|c}) \cdot g(\theta_{c|t}) \cdot g(\theta_t)$. Let $x = (f,t)$ be an observation from a multinomial distribution. Let $F(\theta|x) = g(\theta) \cdot L(x|\theta)$, where $g(\theta)$ defines a prior distribution on the parameters and $L(x|\theta)$ is the likelihood of the (f,t) samples (as in Section 2.2). We want to compute the MAP estimate

$$\theta = argmax_\theta\ F(\theta|x) \tag{12}$$

As $g(\theta_t)$ is a constant function, it does not influence the maximization, and we leave it out in the following estimations. Let

$$F(\theta|x) = (\Pi_{c,f}\ {\theta_{f|c}}^{\alpha_f^c}) \cdot (\Pi_{t,c}\ {\theta_{c|t}}^{\beta_c^t}) \cdot [\Pi_{f,t}\ (\sum_c \theta_{f|c} \cdot \theta_{c|t} \cdot \theta_t)^{n(f,t)}] \tag{13}$$

For maximizing the above function we employ an EM algorithm (similar to the estimation in Section 2.2). The **E-Step** remains the same. The parameter estimates $\theta_{f|c}$ and $\theta_{c|t}$ for the **M-Step** become:

$$\theta_{f|c} = P[f|c] = \frac{\alpha_f^c + \sum_{t \in T} n(f,t) P[c|(f,t)]}{\sum_{f' \in F} (\alpha_{f'}^c + \sum_{t \in T} n(f',t) P[c|(f',t)])} \tag{14}$$

$$\theta_{c|t} = P[c|t] = \frac{\beta_c^t + \sum_{f \in F} n(f,t) P[c|(f,t)]}{\sum_{c' \in C} (\beta_{c'}^t + \sum_{f \in F} n(f,t) P[c'|(f,t)])} \tag{15}$$

Combining the similarity-based estimates with the estimates based on training counts strengthens the model robustness. We show in the next section how the additional flexibility of our model helps learning even when, due to little training, other methods fail.

3 Experiments

3.1 Methodology

The experimental setup is the following: given a large unlabeled collection of documents, the goal is to categorize it into predefined categories. For this purpose, we want to find out how much labeled data is needed for obtaining a reasonable classification accuracy, and what classifier methods perform best. As we most likely have a small amount of labeled (training) data and a large amount of

unlabeled (test) data, the most interesting methods are those that learn over the entire dataset, i.e. transductive techniques. For comparison, we show experiments with both inductive (i.e. methods that learn only from training) and transductive methods. Our results are averaged over 10 repetitions with random splits into training and test data, with the size of the splits ranging from 0.25% training data and 99.75% test data, up to $10\% - 90\%$ training-test splits.

3.2 Test Collections

We evaluate our techniques on three test collections. For all three collections, both stemming and stop-word removal are used.

The first one is the **Reuters-21578 dataset** collected from the Reuters newswire in 1987. Of the 135 potential categories only the most frequent 10 are used (so that enough training/test documents are present) and we keep only documents labeled with a unique topic. This results in a total of 8,024 documents.

The **Amazon** dataset[1] was extracted from `http://www.amazon.com`. It contains editorial reviews of books, organized into categories. From the available categorization, we selected all the editorial reviews for books in *Biological Sciences, Mathematics, and Physics.* This amounts to 5,634 documents.

The **Wikipedia** collection[2] was obtained by a topic-focused crawl of `http://www.wikipedia.org`. The selected topics were *Politics, Computer Science, Physics, Chemistry, Biology, Mathematics, and Geography.* The crawl was started from the main pages of each of these classes and topic-specific words in the anchor text were used as indicators for whether an outgoing link should be followed. The dataset gathered this way contains 5,384 documents. Due to the crawling procedure, this dataset contains a considerable amount of noise. The last two collections differ in nature from the Reuters dataset: their vocabulary is much richer and expressive, the language ambiguity is higher.

3.3 Results

The following experiments show the effect of using explicit knowledge models for transductive inference. As a baseline for comparison the results of a multinomial Naive Bayes (NB) [14], the inductive version of our model coined ILM (Inductive Latent Model) using the MAP estimator of Section 2.4, inductive SVM (ISVM) [13], transductive SVM (TSVM) [12] and the Spectral Graph Transducer (SGT) [11] are shown. Our transductive model is coined TLM (Transductive Latent Model). Based on previous studies [18] about robust evaluation measures for text classification we chose microaveraged F1 as our main performance metric.

In settings with few training data, parameter tuning is infeasible: it is difficult to perform cross-validation or to provide held-out data. For this reason, for all the methods presented, the parameters are kept fixed across experiments. For

[1] Available at `http://www.mpi-sb.mpg.de/~ifrim/data/Amazon.zip`
[2] Available at `http://www.mpi-sb.mpg.de/~ifrim/data/Wikipedia.zip`

Table 1. Reuters. Microaveraged F_1 for different training set sizes.

Training	NB	ILM	ISVM	TSVM	SGT	**TLM**
split0.4	64.5	67.1	71.8	57.6	76.5	**79.7**
split0.5	67.2	70.0	73.5	50.2	79.7	**80.7**
split1	77.2	76.5	81.0	60.8	88.6	**84.1**
split2	83.4	81.7	82.2	72.7	89.9	**85.1**
split5	91.1	90.0	84.9	87.1	92.1	**89.4**
split10	92.9	92.0	88.4	89.4	93.0	**89.6**

Table 2. Wikipedia. Microaveraged F_1 for different training set sizes.

Training	NB	ILM	ISVM	TSVM	SGT	**TLM**
split0.25	72.7	70.4	43.0	32.5	70.5	**79.2**
split0.5	76.0	74.8	61.3	52.0	77.1	**80.5**
split1	77.9	76.9	73.2	63.1	79.3	**80.9**
split2	80.8	80.7	78.9	69.3	81.4	**80.8**
split5	83.6	82.1	83.2	77.8	82.9	**82.8**
split10	85.8	84.5	85.0	81.6	84.3	**84.8**

Table 3. Amazon. Microaveraged F_1 for different training set sizes.

Training	NB	ILM	ISVM	TSVM	SGT	**TLM**
split0.25	77.2	72.7	48.7	64.1	77.8	**83.7**
split0.5	79.1	74.8	63.0	68.0	82.4	**84.9**
split1	81.1	77.9	72.2	75.6	85.0	**85.1**
split2	83.3	81.4	78.2	82.1	86.0	**85.2**
split5	84.2	83.2	83.8	85.8	87.5	**85.5**
split10	85.2	84.9	84.5	86.7	88.1	**85.9**

NB, ILM and TLM, the vocabulary size is fixed to 10,000 terms selected by Mutual Information for NB and ILM and $tf \cdot idf$ rank for TLM. Due to our bootstrapping mapping, our *enhanced* EM algorithm converges very fast, in one or two iterations; for TLM we set the number of EM iterations to 1. As suggested by [12] we do not use any feature selection for SVMs. We use SVMLight [13] with linear kernels and default parameter settings. The parameters for SGT are fixed to the values found in [11] to give the best results on the Reuters dataset: $c = 3200$, $d = 80$ and $k = 800$; no feature selection is done.

As mentioned in Section 2.3, the concept-topic mapping might be poorly estimated from sparse training. We propose solving this problem by automatically querying Wikipedia as background knowledge for improving the bag-of-words context of a given topic, prior to computing the initial topic-concept mapping. For a fair comparison, we give all the other methods the same information as used by TLM. Directly adding the background knowledge to training (e.g. we add

the *Coffee* page from Wikipedia as training for the *Coffee* topic from Reuters) sometimes decreases the accuracy of the other methods (due to variation in vocabulary and data distribution), and sometimes increases the accuracy, particularly for sparse training. We report the best results for all the other methods, so as to give them the best possible advantage against our TLM method.

Table 1 shows results for the Reuters dataset. Using only 33 labeled documents (0.4% of the total dataset), TLM improves the microaveraged F1 from 71.8% for ISVM and 76.5% for SGT, to 79.7%. In our experiments with this dataset, transductive SVM performed considerably worse than inductive SVM for small training sets. This result is different from the one in [12]. We suspect TSVM is particularly sensitive to parameter tuning, thus using default parameters could affect the results. We also note that even if SGT uses its best parameter setting on Reuters, TLM outperforms it for small training data.

Table 2 gives results for the Wikipedia dataset. For the interesting case of little training, TLM outperforms all other methods by a significant margin. The results are particularly impressive for very small training sets, when our model proves to be robust in spite of the little supervised information available. In most of our experiments NB is better than SVM for small training sets; this result is in compliance with existing text classification literature [15], [17].

Table 3 shows results for the Amazon dataset. The results show the same trend: for little training TLM outperforms all other methods. For small training sets (split0.25: 14 documents) TLM outperforms SGT by 6%; for training data large enough (split2: 113 documents) SGT becomes slightly better.

In order to understand the mutual benefits of the bootstrapping mapping and the EM algorithm, we studied the effect of the similarity-based mapping on the classification results, as compared to the results after applying one EM iteration (using maximum likelihood estimates, not MAP). On Reuters, the mapping results were considerably improved by using EM iterations. This can be explained by the nature of this dataset: topics overlap heavily at concept level, thus the mapping alone cannot discriminate well among topics. For both Amazon and Wikipedia the situation differed. For sparse training, employing EM iterations degraded a bit the results obtained with the bootstrapping mapping. This problem of bootstrapping mapping versus EM iterations' quality is largely solved by combining the two estimates in a MAP estimate as shown in Section 2.4.

Running time. The most time consuming part for our method is the mapping of features to concepts which takes about 20 minutes for 10,000 features. Due to our bootstrapping mapping, EM converges very fast, in one or two iterations. Classifying the entire collection is done in a few seconds.

4 Conclusion

This paper has introduced a generative model and a parameter learning algorithm for transductive text classification with explicit background knowledge. In contrast to previously proposed latent semantic models, our approach has explicit concepts associated with its latent variables, which enables it to bootstrap

the EM-based parameter estimation procedure in a highly efficient and effective manner and provides the flexibility for learning from the unlabeled part of the corpus or even additional background corpora. Our extensive experiments have shown significant improvements of classification accuracy, in comparison to state-of-the-art techniques like Spectral Graph Transducer and Transductive as well as Inductive Support Vector Machines. The gains are most pronounced for small training sets, a typical and inevitable situation in many real-life text mining applications.

Acknowledgments. We would like to thank Thomas Hofmann, Deepak Ajwani and the anonymous reviewers for their helpful suggestions.

References

1. Bennet, K.: Combining support vector and mathematical programming methods for classification. Advances in Kernel Methods. MIT-Press (1999)
2. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. NIPS (2002)
3. Bloehdorn, S., Hotho, A.: Text classification by boosting weak learners based on terms and concepts. ICDM (2004)
4. Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. ICML (2001)
5. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufman Publishers (2003).
6. Deerwester, S., Dumais, S.T., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6) (1990)
7. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1999)
8. Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Machine Learning 42(1) (2001)
9. Ifrim, G., Theobald, M., Weikum, G.: Learning word-to-concept mappings for automatic text classification. Learning in Web Search Workshop, ICML (2005)
10. Ifrim, G.: A Bayesian Learning Approach to Concept-Based Document Classification. Master Thesis. http://www.mpi−inf.de/∼ifrim/publications/ (2005)
11. Joachims, T.: Transductive learning via spectral graph partitioning. ICML (2003)
12. Joachims, T.: Transductive inference for text classification using Support Vector Machines. ICML (1999)
13. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. ECML (1998)
14. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. AAAI-98 Workshop on "Learning for Text Categorization (1998)
15. Ng, A., Jordan, M.: On discriminative versus generative classifiers: A comparison of logistic regression and naive bayes. NIPS (2001)
16. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning (39) (2000)
17. Rennie, J.: Tackling the poor assumptions of naive bayes. ICML (2003)
18. Sebastiani, F.: Machine learning in automated text categorization. ACM (2002)
19. Scott, S., Matwin, S.: Feature engineering for text classification. ICML (1999)
20. Vapnik, V.: Statistical learning theory. Wiley (1998)
21. Zhang, T., Oles, F.J.: A probability analysis on the value of unlabeled data for classification problems. ICML (2000)

Exploring Multiple Communities with Kernel-Based Link Analysis

Takahiko Ito[1], Masashi Shimbo[1], Daichi Mochihashi[2], and Yuji Matsumoto[1]

[1] Graduate School of Information Science Nara Institute of Science and Technology
{takahi-i, shimbo, matsu}@is.naist.jp
[2] ATR Spoken Language Communication Research Laboratories
daichi.mochihashi@atr.jp

Abstract. We discuss issues raised by applying von Neumann kernels to graphs with multiple communities. Depending on the parameter setting, Kandola et al.'s von Neumann kernels can identify not only nodes related to a given node but also the most important nodes in a graph. However, when von Neumann kernels are biased towards importance, top-ranked nodes are the important nodes in the dominant community of the graph irrespective of the communities where the target node belongs. To solve this "topic-drift" problem, we apply von Neumann kernels to the weighted graphs (community graph), which are derived from a generative model of links.

1 Introduction

Link analysis techniques are useful for mining knowledge from graph-structured data such as WWW and citation networks. Researchers have been trying to establish measures for evaluating the importance of individual nodes (documents) in a graph, and PageRank and HITS [1] are the two most popular 'global importance' measures. A different type of link analysis measures, the 'relatedness' between graph nodes, has been studied in bibliometrics. Co-citation coupling [2] is a classical measure of relatedness still widely used.

Our previous work [3] showed that it is possible to define a 'mixture' between the HITS global importance and co-citation relatedness through the parameterization of (von) Neumann kernels [4]. These kernels enable to compute the importance of nodes relative to individual 'root' nodes, where the degree of relativity is controlled by the parameter of the kernels.

The Neumann kernels enjoy the properties of HITS, while at the same time they inherit the problem of HITS called 'topic drifts' [5]. This problem is noticeable when the graph consists of multiple communities each addressing different topics. If the Neumann kernels are biased towards importance and applied to a multi-community graph, they assign the highest scores to the nodes in the dominant community irrespective of the root node.

This paper proposes a method to avoid topic drifts when we apply the Neumann kernels to multi-community graphs. To this end, we model the generative process of links, and construct distinct graphs for individual communities. Edges in these graphs have the weights determined by the generation probability of the citation in the respective

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 235–246, 2006.
© Springer-Verlag Berlin Heidelberg 2006

community. Applying Neumann kernels to the community graphs, we can take communities into consideration even when we bias Neumann kernels to importance.

We also discuss the connection between our proposed kernels and the related methods, including pHITS [6] and Hofmann's Fisher kernels based on pLSI [7].

2 Preliminaries

This section reviews the link analysis measures relevant to the subsequent discussion.

Co-citation Coupling Relatedness. Co-citation [2] is the standard methods of computing relatedness (or similarity) between nodes in a citation graph. Co-citation coupling defines relatedness between documents as the number of other documents citing them both. Given the adjacency matrix A of a citation graph, the number of co-citations between nodes i and j is given by the (i, j)-element of the *co-citation matrix* $A^{\mathrm{T}}A$.

HITS Importance. Kleinberg's HITS [1], along with PageRank, is probably one of the most popular methods for evaluating document importance. HITS assigns two scores to each document, called the authority and hub scores. Let A be the adjacency matrix of a citation graph. The HITS algorithm computes the following recursion over $n = 0, 1, \ldots$ starting from $\mathbf{a}_{(0)} = \mathbf{h}_{(0)} = \mathbf{1}$.

$$\mathbf{a}_{(n+1)} = A^{\mathrm{T}}\mathbf{h}_{(n)}/|A^{\mathrm{T}}\mathbf{h}_{(n)}|, \quad \mathbf{h}_{(n+1)} = A\mathbf{a}_{(n+1)}/|A\mathbf{a}_{(n+1)}|.$$

The i-th component of the *authority vector* $\lim_{n\to\infty} \mathbf{a}_{(n)}$ represents the *authority score* of node i. Similarly, the *hub vector* $\lim_{n\to\infty} \mathbf{h}_{(n)}$ gives the *hub scores*. It is well known that under a mild assumption, the authority and hub vectors exist and equal the principal eigenvectors of $A^{\mathrm{T}}A$ and AA^{T}, respectively.

Neumann Kernels. In our previous work [3][8], we analyzed the properties of Kandola et al.'s Neumann kernels [4] as a link analysis measure.

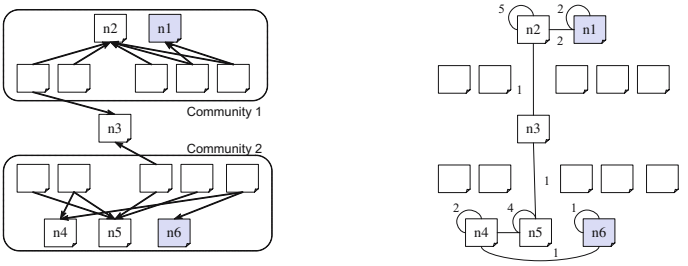
The *Neumann kernel* matrices $\hat{K}_\gamma$ and $\hat{M}_\gamma$, with a parameter γ ($0 \leq \gamma < 1$) called *diffusion factor*, are defined by the following equations.

$$\hat{K}_\gamma = \sum_{n=1}^{\infty} (\frac{\gamma}{\lambda})^{n-1} \left(A^{\mathrm{T}}A\right)^n, \quad \hat{M}_\gamma = \sum_{n=1}^{\infty} (\frac{\gamma}{\lambda})^{n-1} \left(AA^{\mathrm{T}}\right)^n \tag{1}$$

Here, λ represents the dominant eigenvalue of a nonnegative symmetric matrix $A^{\mathrm{T}}A$.

Eq. (1) shows that the Neumann kernel matrix $\hat{K}_\gamma$ (or $\hat{M}_\gamma$) is a weighted sum of $(A^{\mathrm{T}}A)^n$ (or $(AA^{\mathrm{T}})^n$) over $n = 1, 2, \ldots$. Given that the (i, j)-element of the term $(A^{\mathrm{T}}A)^n$ represents the number of paths of length n between nodes i and j in the co-citation graph, we see that each element of the kernel matrix equals the weighted sum of the number of paths between nodes.

We showed in [3] that Eq. (1) can be interpreted as the mixture of co-citation relatedness and HITS importance. As a special case, the Neumann kernels $\hat{K}_\gamma$ subsume co-citation at $\gamma = 0$. At the ceiling of $\gamma \simeq 1$, on the other hand, the rankings induced by any rows of the $\hat{K}_\gamma$ are identical to the HITS importance ranking.



(a) Citation graph with multi-communities (b) Co-citation graph derived from Fig 1 (a)

Fig. 1. A citation graph with multi-communities, and the induced co-citation graphs

3 Topic Drift Problem and Neumann Kernels

HITS is known to suffer from the problem called 'topic drifts' [5]. If applied to a graph with multiple communities[1], HITS assigns the highest scores to the documents in the dominant community of the graph. It follows that the highest scores are assigned to documents unrelated to user's interest, if the topic of dominant community is unrelated to the queries from users.

Consider the graph of Fig 1 (a), which contains two communities. The HITS authority ranking of this graph is $n_2 > n_1 > n_3 > n_5 > n_4 > n_6$, and we see that ranks of documents in Community 2, namely n_4, n_5 and n_6, are uniformly lower than those of documents in Community 1 (n_1 and n_2).

The Neumann kernels reduces to HITS when $\gamma \simeq 1$. This also means that they inherit the issue of topic drifts from HITS. Fig 1 (b) depicts the co-citation graph induced by the graph of Fig 1 (a). The Neumann kernels matrix $\hat{K}_{0.99}$ is shown below.

$$\hat{K}_{0.99} = \begin{pmatrix} 108.53 & 225.98 & 59.64 & 7.16 & 29.30 & 1.36 \\ 225.98 & 477.37 & 127.64 & 15.33 & 62.70 & 2.90 \\ 59.64 & 127.64 & 37.87 & 5.30 & 21.67 & 1.00 \\ 7.16 & 15.33 & 5.30 & 5.16 & 7.34 & 2.17 \\ 29.30 & 62.70 & 21.67 & 7.34 & 23.74 & 1.39 \\ 1.36 & 2.90 & 1.00 & 2.17 & 1.39 & 1.60 \end{pmatrix}. \quad (2)$$

Only the sub-matrix of the kernel matrix for documents n_1 through n_6 is shown. The remaining rows and columns are omitted because these rows and columns correspond to the isolated nodes in Fig 1 (b) and hence all their elements are 0 in the matrix.

The (i, j)-element of this matrix represents the importance of j-th document relative to the i-th document. For example, relative to document n_3 (third row), document n_2 is the most important document because the (3,2)-element (127.64) is the largest in the third row of the matrix.

[1] To the best of our knowledge, there seems to be no consensus on what constitutes 'community' in the literature. Roughly speaking, a 'community' in this paper is a set of documents in a graph citing more documents within the community than those outside.

Now let us focus on the 6-th row in the matrix. In this row, n_2 in Community 1 has the largest value. However, this ranking for n_6 is different from our intuition. The importance score of n_5, a node in the same community as n_6, should be higher than n_2, because the community in n_2 is different from that of n_6.

If we increase a diffusion factor (e.g., $\gamma = 0.999$), the ranking relative to each document will eventually be identical to that of the HITS authority ranking and determined irrespective of the community where the document belongs.

To prevent the importance rankings from diverting away from the topic (or community) users are interested in, Kleinberg [1] proposed to first extract documents that include query terms, and apply the HITS algorithm to the subgraph of the extracted documents.

We may also apply the Neumann kernels to the subgraph of documents containing query terms. However, depending on the type of data (e.g., citation networks), the document contents are not always available. Moreover, Bharat et al. [5] pointed out that there can be a discrepancy between queries and the topics of high-ranked documents, even if HITS is applied to the subgraph of documents containing query terms.

4 Proposed Method

To alleviate topic drifts in the Neumann kernels, we model the generative process of citations, and apply the Neumann kernels to the weighted graphs induced by the model. Unlike Kleinberg's solution, document contents are not used to derive these graphs.

4.1 Generative Model for Citations

Probabilistic Latent Semantic Indexing [9] (pLSI) is a method of modeling the generative process of documents. In pLSI, the joint probability between document d_i and word w_j is given by the following equation.

$$p(d_i, w_j) = \sum_{t=1}^{N} p(t)p(d_i|t)p(w_j|t) \tag{3}$$

where $t \in \{1, 2, \ldots, N\}$ represents a hidden *topic* of documents. The maximum likelihood parameters for $p(t)$, $p(d_i|t)$ and $p(w_j|t)$ are estimated by the EM algorithm.

Cohn et al. [6] modeled the generative process of citations in a similar manner to pLSI. Their model computes the generation probability of citations by using citations in place of words in Eq.(3). Thus, the probability that document i cites j is

$$p(d_i, c_j) = \sum_{t=1}^{N} p(t)p(d_i|t)p(c_j|t)$$

where d_i represents a citation emanating from document i, and c_j represents a citation to document j.

They also proposed to use $p(c_j|t)$, which is the generative probability of a citation within Community t, as the importance of document j in Community t. Although this

method (pHITS) gives the importance rankings by taking the communities into account, it cannot compute the relatedness or relative importance among documents such as those given by the Neumann kernels.

4.2 Applying Neumann Kernels to Community Graphs

In this section, we propose a method to solve the topic drift problem in the Neumann kernels. The proposed method maintains the property of the Neumann kernel as a mixture of importance and relatedness, and also takes communities into consideration, even when it is biased towards importance. This method consists of three steps.

Step 1. Apply pLSI to the citation graph to obtain $p(t|d_i, c_j)$, which is the probability that a citation from document i to another document j is made in the context of community t ($t = 1, \ldots, N$).

Create *community graph* G_t for $t = 1, \ldots, N$, with the same vertex set as the original citation graph, but assign the probability $p(t|d_i, c_j)$ as the weight to the edge from i to j. Thus the adjacency matrix of the t-th community graph $A_t = A(G_t)$ is a square matrix with (i,j)-element of $A(G_t) = p(t|d_i, c_j)$ if $(i, j) \in E$, and 0 otherwise.

Step 2. For each t, apply the Neumann kernels to the co-citation matrix $A_t^T A_t$. The Neumann kernel for community t is given by the following equation.

$$\hat{K}_{t,\gamma} = \sum_{n=1}^{\infty} (\frac{\gamma}{\lambda})^{n-1} \left(A_t^{\mathrm{T}} A_t\right)^n \tag{4}$$

This equation is identical to Eq.(1), except that A_t is used in place of the adjacency matrix of the original citation graph.

Step 3. Finally, sum the Neumann kernels in Eq. (4) over all communities t as follows.

$$R_\gamma = \sum_{t=1}^{N} \hat{K}_{t,\gamma}. \tag{5}$$

This matrix R_γ retains positive semi-definiteness, because the sum of positive semi-definite kernels is still a positive semi-definite kernel [10].

4.3 Relation to Hofmann's pLSI-Based Fisher Kernels

The Fisher kernels [11] are a method to obtain a kernel from generative models. To derive Fisher kernels from a generative model, we need to compute the Fisher score $u(d, \theta)$, the gradient of the log-likelihood function for data d with respect to the parameters θ. Given the Fisher score $u(d, \theta)$, the Fisher kernel is given by following equation.

$$K(d_i, d_j) = u(d_i; \theta)^{\mathrm{T}} I(\theta)^{-1} u(d_j; \theta)$$

Here, $I(\theta)$ is the Fisher information matrix, typically approximated by the unit matrix.

Hofmann [7] proposed a Fisher kernel for computing document similarity based on pLSI. The log-likelihood function of pLSI is given as follows.

$$\log p(d_i) = \sum_j \hat{p}(w_j|d_i) \sum_{t=1}^{N} \log p(w_j|t)p(t|d_i)$$

where $p(w_j|t)$ and $\hat{p}(w_j|d)$ respectively represent the probability that word w_j is generated from topic t, and the empirical probability of word w_j in document d. Hofmann computed the derivatives of the log-likelihood function of pLSI wrt parameters $\rho_{jt} = 2\sqrt{p(w_j|t)}$ and $\rho_t = 2\sqrt{p(t)}$ to yield two types of Fisher kernels. The derivatives wrt ρ_{jt} is given by

$$\frac{\partial \log p(d_i)}{\partial \rho_{jt}} = \frac{\hat{p}(w_j|d_i)p(t|d_i, w_j)}{\sqrt{p(w_j|t)}}.$$

So the resulting Fisher kernel is

$$\bar{K}(d, d') = \sum_{j=1} \hat{p}(w_j|d)\hat{p}(w_j|d') \sum_{t=1}^{N} \frac{p(t|d, w_j)p(t|d', w_j)}{p(w_j|t)}.$$

Analogously, we can define the Fisher kernels for citation networks by replacing word w_j with c_j, namely a citation to document j. The Fisher kernel $\bar{K}$ based on ρ_{jt} in this case can be written as a matrix

$$\bar{K} = \sum_{t=1}^{N} \bar{A}_t^{\mathrm{T}} \bar{A}_t, \tag{6}$$

where the (i,j)-element of $\bar{A}_t$ is $p(t|d_i, c_j)/\sqrt{p(c_j|t)}$ if $(i, j) \in E$, and 0 otherwise.

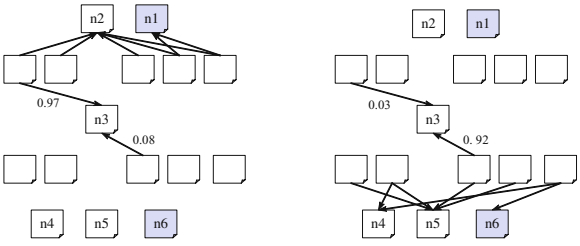
As seen from the term $\bar{A}_t^{\mathrm{T}} \bar{A}_t$ in Eq.(6), this Fisher kernel essentially computes the sum of (reweighted[2]) co-citation graphs for communities $t = 1, \ldots N$. Because computation for each community t is identical to co-citation coupling, the score is zero for any pair of documents not directly co-cited. By contrast, as mentioned in Section 4.2, our proposed kernels (Eq.(5)) compute the weighted sum of all *paths* between documents, so they assign non-zero weights to any document pair as long as they are connected in a community graph; see the discussin on the Neumann kernels in Section 2.

Hofmann used another Fisher kernel based on parameter ρ_t to compensate for this problem. This kernel is defined as

$$\tilde{K}(d, d') = \sum_{t=1}^{N} p(t|d)p(t|d')/p(t).$$

However, this kernel does not have a clear interpretation as a link analysis measure, since this equation does not involve citation c.

[2] A subtle difference between the graphs induced by $\bar{A}_t$ and A_t is that the edge weights in the former are reweighted by $\sqrt{1/p(c_j|t)}$.



(a) community graph 1 (b) community graph 2

Fig. 2. Two community graphs induced by the graph of Fig 1 (a). Edges without weight labels mean their weights equal to 1.0.

4.4 Example

The three steps of our proposed methods (Section 4.2) are demonstrated with a graph of Fig 1 (a). We set the parameter γ for Neumann kernels to 0.99, and the hyperparameter (the number of latent communities) N to 2 in pLSI. In the first step, we apply pLSI to Fig 1 (a) and obtain the two community graphs shown in Fig 2 (a), (b).

In Step 2, the Neumann kernels are applied to each community graph. Because the nodes n_1 and n_2, the most important nodes in terms of HITS authority ranking, are not connected to n_4, n_5 and n_6 in the two community graphs, the scores for n_1 and n_2 relative to the latter nodes are 0 in the respective Neumann kernels.

In Step 3, our proposed kernels (Eq. (5)) sum the Neumann kernel matrices over communities 1 and 2. As a result, the ranking for documents that only have citations within a community graph is biased towards the importance in that community. The ranking relative to the document n_3 located between the two communities is a mixture the importance in two communities.

At $\gamma = 0.99$, the final kernel $R_{0.99}$ is given as follows.

$$R_{0.99} = \begin{pmatrix} 115.04 & 235.81 & 43.56 & 0.00 & 0.00 & 0.00 \\ 235.81 & 489.90 & 91.63 & 0.00 & 0.00 & 0.00 \\ 43.56 & 91.63 & 40.82 & 37.59 & 90.96 & 10.18 \\ 0.00 & 0.00 & 37.59 & 69.38 & 157.23 & 20.07 \\ 0.00 & 0.00 & 90.96 & 157.23 & 375.79 & 42.60 \\ 0.00 & 0.00 & 10.18 & 20.07 & 42.60 & 6.70 \end{pmatrix}. \tag{7}$$

The rankings in the 4-th to 6-th rows are quite different from those of $\hat{K}_{0.99}$ (Eq. (2)). For example, in the 6-th row of $\hat{K}_{0.99}$, the highest score (2.90) is assigned to n_2, which is a node in Community 1, despite that n_6 is in Community 2. By contrast, Eq. (7) assigns the largest value (42.60) to n_5, the node with the most in-links in Community 2. We can see that our proposed kernels are biased towards the importance in the community where each document belongs.

Table 1. HITS authority rankings on community graphs. Columns T, H and P respectively show the index of the communities, HITS authority ranking in the community graph, and the ranking of pHITS.

T	H	P	Title
1	1	1	Building a large annotated corpus of English: the Penn Treebank
	2	3	Statistical decision-tree models for parsing
	3	2	A new statistical parser based on bigram lexical dependencies
	4	6	Unsupervised word sense disambiguation rivaling supervised methods
	5	5	Word-sense disambiguation using statistical models of Roget's categories trained
2	1	2	A stochastic parts program and poun phrase parser for unrestricted text
	2	1	Transformation-based error-driven learning and natural language processing
	3	3	A practical part-of-speech tagger
	4	5	A maximum entropy model for part-of-speech tagging
	5	8	MBT: A memory-based part of speech tagger-generator
3	1	3	Aligning sentences in parallel corpora
	2	1	The mathematics of statistical machine translation: parameter estimation
	3	4	Text-translation alignment
	4	5	A program for aligning sentences in bilingual corpora
	5	6	Char again: a program for aligning parallel texts at the character level
4	1	30	Generating summaries of multiple news articles
	2	25	Empirically designing and evaluating a . . . model for summary generation
	3	10	Generation of extended bilingual statistical reports
	4	33	Practical issues in automatic documentation generation
	5	20	MURAX: A robust linguistic approach for question answering . . .
5	1	1	Attention, intentions, and the structure of discourse
	2	3	Multi-paragraph segmentation of expository text
	3	4	Lexical cohesion computed by thesaural relations as an indicator of the structure of text
	4	7	Combining multiple knowledge sources for discourse segmentation
	5	11	A prosodic analysis of discourse segments in direction-giving monologues

5 Experiments

To evaluate the characteristics of the measures induced by our proposed kernels, we apply them to a bibliographic citation network in the field of natural language processing consisting of 2280 nodes (papers). In the experiments throughout this section, we set the number of communities (N) in pLSI to 5.

5.1 Community Graphs

Before examining the characteristics of the kernels, we check whether communities induced by Step 1 of our method are sensible. We computed the HITS and pHITS rankings for each community graph G_t to elucidate the central topic of each community. The top-5 lists are shown in Table 1.

From the title of the top ranked papers, we see that Community 1 represents a mixture of 'parsing' and 'word sense disambiguation'. Community 2 is the 'part-of-speech tagging' community. Community 3 is on 'machine translation'. Note that 'sentence

Table 2. The top-10 list generated by the plain Neumann kernels for 'Empirical studies in discourse' ($\gamma = 0.001$)

$\hat{K}$	C	H	Title
1	1	771	**Empirical studies in discourse**
2	2	1	Building a large annotated corpus of English: the Penn Treebank
3	2	50	**Attention, intentions, and the structure of discourse**
4	2	76	**Assessing agreement on classification tasks: the Kappa statistic**
5	2	201	**The reliability of a dialogue structure coding scheme**
6	2	604	**Message Understanding Conference (MUC) tests of discourse processing**
7	2	1061	**Effects of variable initiative on linguistic behavior in . . . language dialogue**
8	–	3	Statistical decision-tree models for parsing
9	–	4	A new statistical parser based on bigram lexical dependencies
10	–	96	**Centering: a framework for modeling the local coherence of discourse**
11	–	8	Three generative, lexicalised models for statistical parsing

alignment' is a fundamental technique to statistical machine translation. Community 4 represents the related fields of 'summarization' and 'generation' of documents. Finally, community 5 is concerned about 'discourse processing'. There is a clear similarity between the ranking from HITS and pHITS on each community except community 4[3].

5.2 Comparison

We computed (i) plain Neumann kernels, and (ii) our proposed kernels (Eq. (5)). Each kernel matrix was treated as a ranking method by taking the i-th row vector of the matrix as the score vector for the i-th node (paper). Given the i-th score vector, or the ranking induced thereof, we call the i-th node as the *root paper* of this ranking.

To observe the differenc in the character of these kernels, We compare the top-10 papers relative to a fixed root paper, 'Empirical studies in discourse' by M. A. Walker and Johanna D. Moore, *Computational Linguistics* 23(1):1–12, 1997. This paper was used in our previous work as well, and ranks 14-th in Community 5 by HITS. Another reason we selected this root paper is that it is a paper on discourse processing. Because papers on discourse are not ranked among the top 10 HITS ranking (see column H in Table 4), the ranking for this root paper is prone to the topic drift phenomenon.

Plain Neumann Kernels. Tables 2 ($\gamma = 0.001$), 3 ($\gamma = 0.95$) and 4 ($\gamma = 0.9999$) show the list of top-10 papers induced by the plain Neumann kernels relative to the root paper 'Empirical studies in discourse.' Columns $\hat{K}$, C and H respectively display the rankings induced by the plain Neumann kernels, co-citation coupling, and the HITS authority score. A '$-$' in column C indicates that the paper was not co-cited with the root paper. Tht titles of papers on discourse processing are shown in boldface.

When $\gamma = 0.001$ (Table 2), the majority is formed by the papers on discourse processing. Ranked topmost is the root paper, followed by the six papers co-cited with the root paper, as indicated by column C.

[3] In community 4, the rankings between HITS and pHITS are not similar, but pHITS also ranked paper on summarization and generation topmost.

Table 3. The top-10 list generated by the plain Neumann kernels for 'Empirical studies in discourse' ($\gamma = 0.95$)

$\hat{K}$	C	H	Title
1	2	1	Building a large annotated corpus of English: the Penn Treebank
2	–	3	Statistical decision-tree models for parsing
3	–	2	A stochastic parts program and noun phrase parser for unrestricted text
4	–	4	A new statistical parser based on bigram lexical dependencies
5	2	50	**Attention, intentions, and the structure of discourse**
6	1	771	**Empirical studies in discourse**
7	–	8	Three generative, lexicalised models for statistical parsing
8	2	76	**Assessing agreement on classification tasks: the Kappa statistic**
9	–	6	Word-sense disambiguation using statistical models of Roget's categories trained
10	–	5	Unsupervised word sense disambiguation rivaling supervised methods

Table 4. The top-10 list generated by the plain Neumann kernels for 'Empirical studies in discourse' ($\gamma = 0.9999$)

$\hat{K}$	C	H	Title
1	2	1	Building a large annotated corpus of English: the Penn Treebank
2	–	2	A stochastic parts program and noun phrase parser for unrestricted text
3	–	3	Statistical decision-tree models for parsing
4	–	4	A new statistical parser based on bigram lexical dependencies
5	–	5	Unsupervised word sense disambiguation rivaling supervised methods
6	–	6	Word-sense disambiguation using statistical models of Roget's categories trained
7	–	7	The mathematics of statistical machine translation: parameter estimation
8	–	8	Three generative, lexicalised models for statistical parsing
9	–	9	Transformation-based error-driven learning and natural language processing
10	–	10	Integrating multiple knowledge sources to disambiguate word sense

In Table 3, we show the ranking at $\gamma = 0.95$ as a measure midway between relatedness and importance. Although the parameter value $\gamma = 0.95$ might seem too biased towards 1.0 (HITS importance), transition from relatedness to importance occurs rather late in the parameter range (typically in the range of $0.9 < \gamma < 1.0$) [3]. As a result, the ranking at $\gamma = 0.5$, for example, is mostly identical to $\gamma = 0.1$.

Only three papers on discourse remain in Table 3. These three eventually fall out of top 10 at $\gamma = 0.9999$ (Table 4). The ranking at $\gamma = 0.9999$ is identical to HITS importance ranking. We also sampled several other parameter points between $\gamma = 0.001$ and 0.9999. None of these ranking lists included discourse papers other than those appeared in Table 2 for $\gamma = 0.001$.

Proposed Kernels. Tables 5 ($\gamma = 0.001$), 6 ($\gamma = 0.95$) and 7 ($\gamma = 0.9999$) show the lists of top-10 papers induced by the proposed 'community-aware' kernels R_γ relative to 'Empirical studies in discourse.' In these tables, column R shows the rankings induced by the proposed kernels.

At $\gamma = 0.001$ (Table 5), the ranking is similar to that of the plain Neumann kernels (Table 2) and most of top ranked papers are on discourse processing.

Table 5. The top-10 list generatd by the proposed kernels for 'Empirical studies in discourse' ($\gamma = 0.001$)

R	C	H	Title
1	1	771	**Empirical studies in discourse**
2	2	201	**The reliability of a dialogue structure coding scheme**
3	2	76	**Assessing agreement on classification tasks: the Kappa statistic**
4	2	1061	**Effects of variable initiative on linguistic behavior in ... language dialogue**
5	2	50	**Attention, intentions, and the structure of discourse**
6	2	1	Building a large annotated corpus of English: the Penn Treebank
7	2	604	**Message Understanding Conference (MUC) tests of discourse processing**
8	–	96	**Centering: a framework for modeling the local coherence of discourse**
9	–	374	A trainable document summarizer
10	–	60	Evaluating a focus-based approach to anaphora resolution

Table 6. The top-10 list generated by the proposed kernels for 'Empirical studies in discourse' ($\gamma = 0.95$)

R	C	H	Title
1	1	771	**Empirical studies in discourse**
2	2	201	**The reliability of a dialogue structure coding scheme**
3	2	76	**Assessing agreement on classification tasks: the Kappa statistic**
4	2	50	**Attention, intentions, and the structure of discourse**
5	2	1061	**Effects of variable initiative on linguistic behavior in ... language dialogue**
6	2	1	Building a large annotated corpus of English: the Penn Treebank
7	–	96	**Centering: a framework for modeling the local coherence of discourse**
8	–	61	**Multi-paragraph segmentation of expository text**
9	–	77	**Lexical cohesion computed by thesaural relations ...**
10	–	115	**Combining multiple knowledge sources for discourse segmentation**

Table 7. The top-10 list generated by the proposed kernels for 'Empirical studies in discourse' ($\gamma = 0.9999$)

R	C	H	Title
1	2	50	**Attention, intentions, and the structure of discourse**
2	–	61	**Multi-paragraph segmentation of expository text**
3	–	77	**Lexical cohesion computed by thesaural relations ...**
4	–	115	**Combining multiple knowledge sources for discourse segmentation**
5	–	198	**A prosodic analysis of discourse segments in direction-giving monologues**
6	2	76	**Assessing agreement on classification tasks: the Kappa statistic**
7	–	150	**An automatic method of finding topic boundaries**
8	–	162	**Text segmentation based on similarity between words**
9	–	317	**Intention-based segmentation: Human reliability and correlation with ...**
10	–	340	**Replicability of transaction and action coding in the map task corpus**

At $\gamma = 0.95$ (Table 6), we see the increase in the number of discourse papers, which is contrastive to the plain Neumann kernel with $\gamma = 0.95$ (Table 3) listing only three discource papers.

The ranking list in Table 7 for $\gamma = 0.9999$ consists solely of papers on discourse processing and text segmentation, a subtask of discourse processing. The root paper and most of the papers co-cited with the root are not on this list, but the top 5 papers match the most important papers for Community 5 (Table 1).

Fron these results, we see taht our proposed kernels did not suffer from the topic drift problem for this root paper; when γ is increased, they tend towards importance within the community where the root paper belongs. By contrast, plain Neumann kernels often assigned higher scores to papers in other communities, such as 'A new statistical parser based on bigram lexical dependencies'; see Tables 2, 3 and 4.

6 Conclusions

We constructed a citation graph for each community using a technique similar to pLSI and pHITS. Applying Neumann kernels to each community graph, we can rank documents by taking the community of each individual document into consideration.

The technique proposed in this paper can be extended with other latent topic models beside pLSI. We are planning to apply Latent Dirichlet Allocation [12] to construct community graphs.

References

1. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM (1999) 604–632
2. Small, H.: Co-citation in the scientific literature: a new measure of the relationship between two documents. J. American Society for Information Science **24** (1973) 265–269
3. Ito, T., Shimbo, M., Kudo, T., Matsumoto, Y.: Application of kernels to link analysis. In: Proc. 11th ACM SIGKDD. (2005) 586–592
4. Kandola, J., Shawe-Taylor, J., Cristianini, N.: Learning semantic similarity. In: NIPS 15. (2002)
5. Bharat, K., Henzinger, M.R.: Improved algorithms for topic distillation in a hyperlinked enviornment. In: Proc. 21st ACM SIGIR Conference. (1998)
6. Cohn, D., Chang, H.: Learning to probabilistically identify authoritative documents. In: Proc. 18th International Conference of Machine Learning. (2001)
7. Hofmann, T.: Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In: NIPS 12. (2000) 914–920
8. Shimbo, M., Ito, T.: Kernels as link analysis measures. In Cook, D., Holder, L., eds.: Mining Graph Data. John Wiley & Sons (2006) In press.
9. Hofmann, T.: Probabilistic latent semantic indexing. In: Proc. 22th ACM SIGIR Conference. (1999) 50–57
10. Haussler, D.: Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz (1999)
11. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: NIPS 11. (1998)
12. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. In: NIPS 14. (2001)

Distribution Rules with Numeric Attributes of Interest*

Alípio M. Jorge[1], Paulo J. Azevedo[2], and Fernando Pereira[1]

[1] LIACC, Faculty of Economics, University of Porto, Portugal
amjorge@liacc.up.pt
[2] Departamento de Informática, University of Minho, Portugal
pja@di.uminho.pt

Abstract. In this paper we introduce distribution rules, a kind of association rules with a distribution on the consequent. Distribution rules are related to quantitative association rules but can be seen as a more fundamental concept, useful for learning distributions. We formalize the main concepts and indicate applications to tasks such as frequent pattern discovery, sub group discovery and forecasting. An efficient algorithm for the generation of distribution rules is described. We also provide interest measures, visualization techniques and evaluation.

1 Introduction

Learning and discovering probability distributions is an important and difficult problem in statistics, machine learning and data mining [12]. Machine learning has focused particularly on learning conditional probabilities of one target variable y (either numerical or categorical) with respect to a set of input variables X. However, the output of a learning algorithm is typically reduced to associating the most adequate value of y to each combination of values of the variables in X. This is the case in regression, classification and association discovery. Learning whole distributions goes beyond point estimation. In this paper, we approach the problems of discovering and presenting important conditional distributions of a target variable with respect to a set of input variables. Our approach is based on association rule discovery [1].

Association rules (AR) are highly legible chunks of knowledge that can be discovered from data. On top of that, the process for generating association rules is efficient enough to deal with very large databases, and the intended result is very well defined and free of heuristics. Although devised mainly for descriptive purposes, AR can also be useful in classification [14], clustering [10], regression [17], recommendation and subgroup discovery [11].

Typically, algorithms for the discovery of AR deal with categoric attributes only. Srikant [19] proposed a specific approach for the discretization of numerical attributes bearing in mind the descriptive aim of AR. In predictive tasks

* Supported by POCI/TRA/61001/2004 Triana Project (Fundação Ciência e Tecnologia), FEDER e Programa de Financiamento Plurianual de Unidades de I & D.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 247–258, 2006.
© Springer-Verlag Berlin Heidelberg 2006

such as regression, [17] or classification [14] the independent numeric variables can be discretized using the supervised discretization MDL based algorithm [8]. Avoiding pre-discretization, Fukuda *et al.* [9] proposed an algorithm for handling pairs of numeric attributes on the LHS of association rules. Aumann and Lindell [2] introduced *Quantitative Association Rules (QAR)*, where a frequent itemset (on the LHS of the rule) is associated with a statistical summary of a numeric attribute of interest(on the RHS). Numeric attributes appearing on the LHS are pre-discretized. Other authors have meanwhile improved some aspects of the original QAR [20,21], in a different direction from the work proposed here.

To learn and discover distributions we propose *distribution rules* (DR). These associate a frequent itemset with an empirical distribution of a numeric attribute of interest without any loss of information. Distribution rules can be used in descriptive data mining tasks with the advantage of avoiding pre-discretization of the numeric variable of interest. We provide an efficient algorithm that discovers distribution rules and describe how to filter interesting rules, using the statistical distribution of Kolmogorov-Smirnov. Distribution rules can be easily visualized as frequency polygons and viewed by a domain expert or data analyst. Besides, DRs can also potentially be used in a predictive setting, and are not fundamentally limited to numeric properties of interest.

2 Distribution Rules

Definition. A *distribution rule* (DR) is a rule of the form $A \rightarrow y = D_{y|A}$, where A is a set of items as in a classical association rule, y is a property of interest (the target attribute), and $D_{y|A}$ is an empirical distribution of y for the cases where A is observed. This attribute y can be numerical or categorical. $D_{y|A}$ is a set of pairs $y_j/freq(y_j)$ where y_j is one particular value of y occurring in the sample and $freq(y_j)$ is the frequency of y_j for the cases where A is observed. $\diamond$

In this paper we will assume y is a numeric variable. Nevertheless, the concept of distribution rules is extended for categorical attributes as well. The attributes on the antecedent are either categorical or are discretized as in [8].

Example. Suppose we have clinical data describing habits of patients and their level of cholesterol. The distribution rule $smoke \wedge young \rightarrow chol = \{180/2, 193/4, 205/3, 230/1\}$ represents the information that, of the young smokers on the data set, 2 have a *cholesterol* of 180, 4 of 193, 3 of 205 and 1 of 230. This information can be represented graphically, for example, as a frequency polygon. The attribute *chol* is the property of interest.$\diamond$

Given a dataset S, the task of *distribution rule discovery* consists in finding all the DR $A \rightarrow y = D_{y|A}$, where A has a support above a determined mininum σ_{min} and $D_{y|A}$ is statistically significantly different (w.r.t. a pre-defined threshold) from the default distribution $D_{y|\emptyset}$. The default distribution is the one obtained with all the values of y for the whole dataset.

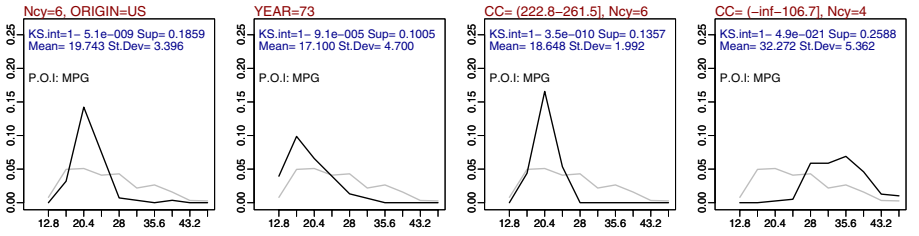


Fig. 1. Graphical representation of one distribution rule for the dataset auto-mpg

2.1 Presentation and Visualization

Although distribution rules can be output as text, the length of the empirical distribution is normally too long to be readable in practice. Since the consequent of one distribution rule is an empirical distribution, it can be represented as a frequency polygon. In Figure 1 we can see 4 rules obtained from the dataset auto-mpg [16]. The antecedent of each rule (e.g., the leftmost) is displayed as the main title. Some selected measures of the distribution and the name of the property of interest (P.O.I.: MPG) are shown within the plot. The x axis is the domain of the P.O.I. and the y axis the estimated probability density. The polygon is drawn by binning the domain of the P.O.I. into a given number of intervals (default 10) with equal width w. For each interval I, the pair x, y is plotted. The value of x is the lower limit of the interval and $y = freq_I/(freq_r * w)$, where $freq_I$ is the number of values in I and $freq_r$ is the total number of values of the P.O.I. covered by the rule.

The distribution for the set of cases that satisfy the condition is shown in black, and the default distribution for the whole population is shown in grey. For the distribution rule shown in Figure 1 we can see that cars with 6 cylinders built on the US tend to make less miles per galon than the whole set of cars. For those cars, the values of MPG are very concentrated around 20. Nevertheless, we can see that there are some economic cars in this group because of the right tail of the black curve. The interest of this rule is shown as KS.int, the complement to 1 of the Kolmogorov-Smirnov test p-value as explained in the following section.

Alternatively, the empirical distribution could be represented by a parametric distribution curve (e.g., Normal), or a boxplot. In this paper we adopted the frequency polygon, since it does not require any assumption about the distribution of the P.O.I. and it minimizes the loss of information regarding the distribution.

2.2 Measuring the Interest of DRs

The interest of a discovered pattern can be measured according to objective and subjective criteria [18]. In the case of association rules, objective interest measures typically try to assess how much the observed frequency of the consequent of the rule, under the conditions imposed by the antecedent, deviate from the frequency that would be expected assuming that antecedent and consequent

were independent. This is the case of measures such as *lift* (a.k.a. *interest*), *leverage* or *conviction*[5]. The χ^2 statistical test has also been extensively used for testing the statistical independence between the antecedent and consequent of association rules [15].

In the case of distribution rules, objective interest can be measured by assessing the difference between the distribution of the consequent and a reference distribution. This, in principle, is the distribution of the whole population. The difference between two empirical distributions can be assessed through a statistical goodness of fit test, such as *Kolmogorov-Smirnov* [7].

Definition. Given a set of transactions DS, a property of interest y in DS, and a distribution rule $A \rightarrow D_{y|A}$ obtained from DS, the *KS-interest* of that rule is $1-p$ where p is the p-value of the Kolmogorov Smirnov test for the two empirical distributions $D_{y|A}$ and $D_{y|\emptyset}$. ◇

Given this notion of the interest of a distribution rule, we can filter a set of DR's by selecting the ones with KS-interest above a pre-defined threshold. This threshold can be intuitively set by a data analyst since it has a clear statistical meaning. Although other notions of interest can also be defined using other statistical tests, we will for now focus on the use of the KS test.

3 Using DRs

Distribution rules can be used in descriptive pattern discovery tasks, although they can also be adopted in predictive tasks as well. One immediate advantage of their use in these situations is that it is not required to previously discretize the attribute y.

One way to handle distribution rules is by working with them as regular association rules. In Table 1 we can see a textual representation of one rule discovered for the dataset *Determinants of Wages from the 1985 Current Population Survey in the United States*, a.k.a. *Wages*, also used in [2]. While the antecedent of each of these rules is a frequent itemset, the consequent is a raw distribution. Although the rules can be represented in this textual form, they are internally stored using compact data structures. To present the rule, it is more effective to graphically visualize them, or to summarize them, as for example in [2].

Having obtained a set of distribution rules, these can be presented, sorted and filtered in many different ways. In this paper, we propose one particular graphical multi-plot presentation (Figure 2). In each plot, the default distri-

Table 1. A distribution rule produced for the dataset *Wages*, with min-sup=0.1, min-KS.int=1 − 0.05 and a minimal KS improvement of 0.01

```
Sup=0.118 KS.int=1-0.0085 Mean=10.982 St.Dev=6.333
EDUCATION=(12.5-15.5] & SOUTH=0 & RACE=3
  ->  WAGE={ 3.98/1,4.0/1,4.17/1,4.5/1,4.55/1,4.84/1,5.0/1,5.62/1,5.65/1,5.8/1,6.0/1,6.25/4,7.14/1,7.5/1,7.67/1,7.7/1,7.96/1,
             8.0/2,8.4/1,8.56/1,8.63/1,8.75/1,8.9/1,9.22/1,9.63/1,9.75/1,9.86/1,10.0/3,10.25/1,10.5/1,10.53/1,10.58/1,10.61/1,
             11.11/1,11.25/2,12.0/1,12.47/1,12.5/4,13.07/1,13.75/1,13.98/1,14.29/1,15.0/1,16.0/1,16.14/1,16.42/1,17.25/1,17.86/1,
             18.5/1,21.25/1,22.5/1,26.0/1,44.5/1 }
```

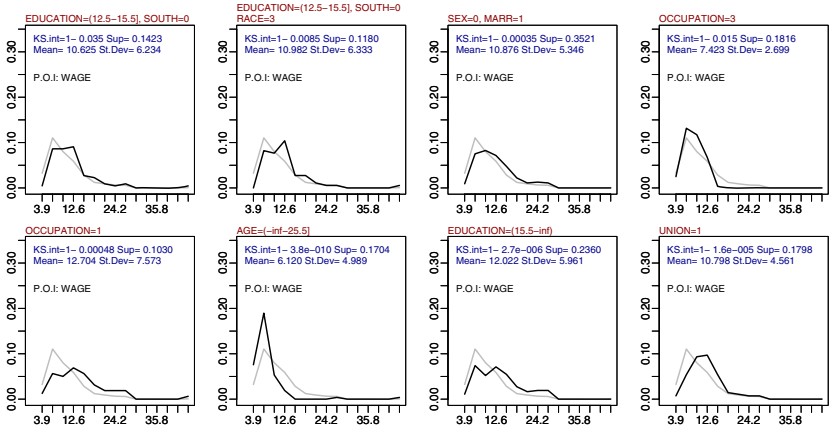


Fig. 2. Multi plot of a set of 8 distribution rules. Each is plotted against the default distribution.

bution is used as a term of comparison and appears in grey. The rules shown are a subset of the 35 rules produced for the dataset *Wages*, obtained with a minimal antecedent support of 0.1 and a min KS-interest of 0.95. We have also applied an improvement filter, as suggested in [4], on the KS-interest. In this case, $improvement(A \rightarrow B)$ can be defined as $\min(\{\text{KS-interest}(A \rightarrow B) - \text{KS-interest}(A_s \rightarrow B) \mid A_s \subseteq A\}$. The minimal KS-interest improvement used in these experiments was 0.01.

The 35 rules can be visualized in mosaic plots of $n \times m$. We show a mosaic of 2×4 with 8 rules selected from the 35. The selection can be done visually by paging the 35 rules in mosaics of $n \times m$. We will refer to the rules on Figure 2 by number from 1 to 8, reading from left to right and from top to bottom. Rules 1 and 2 describe people with 13 to 15 years of education which are not from the South. Their wages distribution is significantly different from the whole population and visibly better although concentrated on the same interval. Rule 2 is a refinement of rule 1 with higher interest. It seems that people with race=3 (white) in the conditions stated before have a slightly but significantly better situation. Rule 3 describes married males, and rules 4 and 5 show that occupation 1 has a wider and higher range of income than occupation 3. Rule 6 shows the impact of age, and rule 8 the positive effect of holding a union membership. Rule 7 indicates that people with higher education have higher wages.

Used in this setting, distribution rules are selected by the Kolmogorov-Smirnov statistic. Improvement enables the elimination of non informative sub rules. The visualization of the distributions gives a broader picture of the subset of data covered by each rule. With these parameters (min KS-int=0.95 and min improvement=0.01) we get very few rules.

Distribution rules can be naturally applied to the data minig task of subgroup discovery [13] both for numeric and categorical properties of interest. An interesting subgroup corresponds to a KS-interesting distribution rule.

Algorithm 1. CAREN-DR Depth First Distribution rules derivation

Input: *minsup*, KS-int = $1 - \alpha$, *DB*
Rules = $\emptyset$;
First database scan (count items)
Build DI = {items of the form $y = v_i$ belonging to property of interest};
Build AI = {antecedent items with count $\geq$ *minsup*} ;
Second database scan (bitmaps mounting)
Mount coverage bitmap for each item in AI and DI;
Compute D_y using DI bitcounting;
foreach *transaction* $t \in DB$ **do**
  Set correspondent bit in each item (in AI and DI) occurring in t;
  Count 2-itemsets occurring in t;
end
`/* (Expansion phase)` `*/`
foreach *frequent item* $i \in$ AI **do**
  Compute $D_{y\,i}$ from *bitmap*(i) and *bitmaps*(DI);
  if $KS(D_y\ , D_{y\,i}) < \alpha$ **then** $Rules = Rules \cup \{i \rightarrow D_{y\,i}\}$;
  foreach *frequent item* $i\ > i$ *(> refers to items ordering)* **do**
    $a = \{i, i\ \}$
    bitmap(a) = *bitmap*(i) $\oplus$ *bitmap*(i);
    if *support*(a) $\geq$ *minsup* **then**
      Compute $D_{y\,a}$ from *bitmap*(a) and *bitmaps*(DI);
      if $KS(D_y\ , D_a) < \alpha$ **then** $Rules = Rules \cup \{a \rightarrow D_{y\,a}\}$;
      $Rules = Rules \cup$ `Expansion`$(a, i\ , D_y\ , \alpha)$;
    end
  end
end
Output: *Rules*

Distribution rules can also potentially be used in predictive tasks such as regression as in [17] or probability density estimation as in [6]. In this paper we have focused on the fundamental concepts and on the processes of generating, filtering and presenting the rules.

4 Rule Generation

A set of distribution rules can be obtained from a given database by computing all the frequent itemsets a not involving the property of interest y. For each frequent itemset a we compute the associated distribution $D_{y|a}$. Counting operations are efficiently implemented through the use of bitmaps.

4.1 Algorithm and Computational Complexity

The algorithm CAREN-DR works by finding frequent itemsets and, simultaneously, their associated p.o.i. distributions. For each antecedent item, a bitmap

```
   Input: (itemset,lastitem,D_y  ,α)
 1 R = ∅;
 2 foreach i ∈ AI , i > lastitem (> refers to items ordering) do
 3     if  ∀ a ∈ itemset  support({a, i}) ≥ minsup then
 4         new = itemset ∪ {i};
 5         bitmap(new) = bitmap(itemset) ⊕ bitmap(i);
 6         support(new) = bitcount(new);
 7         if support(new) ≥ minsup then
 8             Compute D_y new from bitmap(new) and bitmaps(DI);
 9             if KS(D_y  , D_new) < α then R = R ∪ {new → D_new}
10             R = R ∪ Expansion(new, i, D_y  , α);
11         end
12     end
13 end
14 return R
```

Function. `Expansion`(*itemset,lastitem,*$D_{y|\emptyset}$,α)

that represents its coverage is built. Antecedents are formed by depth first expansion. When an item is added to the antecedent to build a new itemset, a new bitmap is calculated (through bit-intersection) and its support can be counted through a bitcounting operation. To help in unfrequent itemset pruning during itemset expansion, the algorithm builds a flat matrix with 2-itemsets counts. Thus, expensive bitcounting operations can be avoided if subsets of the candidate itemsets are not frequent.

For an efficient rule's consequent calculation, each distribution item (the numeric values associated with the p.o.i.) also keeps a bitmap. Deriving a new distribution requires intersection operations between the bitmap of the antecedent itemset and the bitmaps of the distribution items. The algorithm extracts significant rules by performing a Kolmogorov-Smirnov test between each new rule ($D_{y|a}$) and the *a priori* distribution ($D_{y|\emptyset}$).

The algorithm receives as input a minsup for antecedent filtering and an α that is used to set the minimal KS-interest threshold in $1-\alpha$. It can also receive an improvement threshold value. The theoretical complexity of the method is dominated by the complexity of finding frequent itemsets, which is known to be linear on the number of cases. Bitmap operations for the P.O.I. distribution update are also linear on the number of cases.

CAREN-DR algorithm conceptually resembles OPUS-IR algorithm [20], since it also uses a depth-first approach. In fact, with minimal adjustments, our proposal can be easily modified to work in a top-N rules search mode. However, we use bitmaps to represent itemset coverage and to calculate p.o.i. distributions. Our implementation of CAREN-DR is part of the java-based association rule discovery engine CAREN [3]. In relation to the QAR proposal of Aumann and Lindell, our algorithm does not require an extra database scan to compute the distributions associated to each rule. Furthermore our method outputs whole distributions and defines the interest of a rule in terms of comparison of distributions rather than the comparison of means.

Table 2. Description of the datasets used to measure the computation time (upper table). The column #Distinct indicates the number of distinct values of the property of interest (p.o.i.). Times in seconds and number of rules generated for the datasets for different minimal supports (lower table).

Dataset	#Attr	#Records	p.o.i.	#Distinct
mpg	7	398	MPG	129
housing	13	506	MEDV	211
abalone	8	4177	RINGS	28
cal. houses	9	20640	mhousevalue	3842

	Time in seconds				Number of rules generated			
min.sup	MPG	Housing	Abalone	Cal. Houses	MPG	Housing	Abalone	Cal. Houses
0.3	3.202	6.811	12.425	36.993	4	98	4	6
0.2	3.220	7.107	12.313	43.936	15	310	4	22
0.1	3.629	8.790	12.281	56.326	67	1490	41	74
0.05	5.089	13.894	12.790	72.084	240	5848	516	197
0.01	5.643	48.962	15.095	153.867	1369	51185	3158	1867

5 Evaluation

In this section we show how our algorithm CAREN-DR performs on 4 different datasets described in Table 2. The algorithm has been run with different values of minimal support for a minimal KS-interest of 0.95 and with the improvement switch turned off. We can see that the algorithm scales up quite well with the number of examples and the value of minimal support. Table 2 (bottom) shows the times in seconds spent on a Pentium IV, 1.6GHz and 1GB RAM. These times include writing the rules to a csv file (one of the possible output modes). Table 2 (bottom) also shows the number of rules produced per run. We stress, however, that by turning improvement on, the number of rules falls dramatically.

These experiments show that the algorithm is capable of generating a very large number of distribution rules (and writing them as text to disk) in a very reasonable time (51185 rules for *Housing* in 49 seconds). In the case of the dataset *Cal. Houses*, CAREN-DR processes the 20640 cases in 2.5 minutes. Additional experiments with this dataset show that the time spent by CAREN-DR grows practically linearly as the number of examples rises from 5000 to 20000. In another set of experiments we observe that the time spent by the KS-test is also linear w.r.t the size of the distributions.

In our approach, the number of different values of the property of interest is also a source of complexity. However, typical numerical attributes tend to have low numerical precision, thus low variety of values. In the event of having to deal with a high precision attribute, we can round the values to a reasonable number of significant digits. Experimentally, we observe that the p-value of the KS test is robust to rounding to 3 significant digits.

5.1 An Artificial Dataset

In order to test the ability of the KS test to identify interesting rules, we have generated an artificial dataset with 1000 cases. The values of the attributes were chosen so that specific interesting distribution rules should appear. Thus, we have randomly initially generated the values for the p.o.i. y from a N(0,1) distribution.

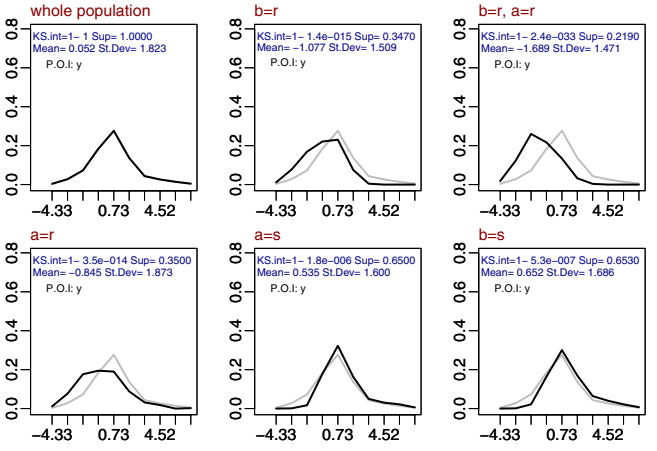


Fig. 3. Distribution rules for the artificial dataset

After that we have randomly assigned values of r or s to the categorical attributes a, b, and c. Whenever a and b had the values r we added to the value of y an extra random value from a distribution N(-2,1). The algorithm CAREN-DR produced the distribution rules shown in Figure 3 with a minimal support of 0.1, and a minimal KS-interest of 0.95. The minimal improvement on KS-interest used was 0.001. If the improvement filter is turned off, some redundant rules appear.

As we can see, only 5 rules have been identified, apart from the *a priori* rule. The condition $a = r \,\&\, b = r$ appears as expected, but also its items separately and their complements. The attribute c does not appear since the distribution of $y|c$ is similar to the distribution of $y|\emptyset$.

6 Case Study

We have applied distribution rules to the analysis of the main causes of delays in trip time duration for buses in a urban centre. This is a real dataset with about 8000 cases describing trips of a specific bus line. The dataset has 16 attributes plus the property of interest TripTime. The numeric attributes of the antecedent have been previously discretized using an implementation of the algorithm of Fayyad and Irani [8]. Since this discretization approach requires a class attribute, it is done with respect to a discretized version of the P.O.I, as in [17]. Afterwards, the P.O.I. is used in its continuous version. We obtain 36 relevant rules with support above 0.05 and KS-interest above 1-1E-05. Improvement is 0.0001.

In Figure 4 we can see a selection of the rules. Most rules have only one condition on the antecedent due to the effect of improvement filtering. We can see for example the difference in the distribution of the time a bus takes to make its route in March (Month=3) and in August (Month=8). Holydays also have a positive impact on trip time (plot 6) . The last two plots show the difference between Sundays and Fridays. The other attributes that appear on Figure 4 are

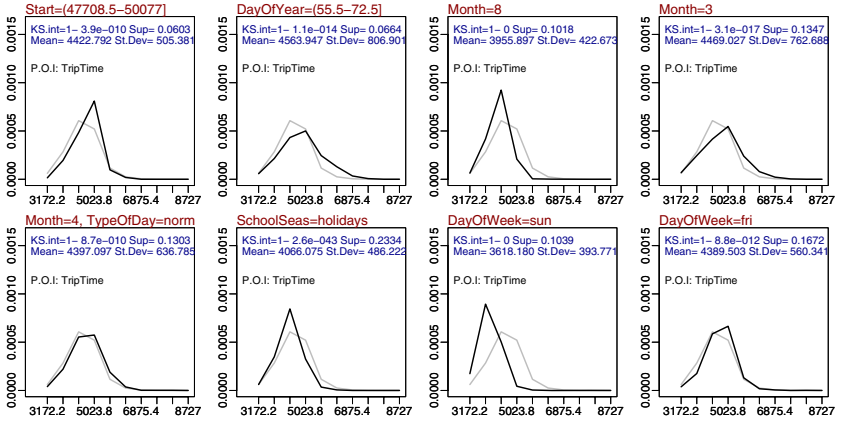


Fig. 4. Distribution rules for the buses dataset

Start, which is the starting time of the bus trip in seconds, DayOfYear, which is the number of days passed since 1st of January of the year being studied, and TypeOfDay, which can have values normal, or bank holiday.

This type of rules are being used to attempt to reduce the costs with personnel, since unpredicted delays often force the bus company management to pay for extra labour time. This way, distribution rules can be used both to give managers indications about the most relevant causes of delay and also enable to predict the probability that TripTime will be higher than a certain threshold.

7 Related Work

Distribution rules are mainly related to learning probability distributions [12], subgroup discovery [13] and quantitative association rules (QAR).

Aumann and Lindell's work on QAR uses a z-test to identify rules significance. As already pointed by Webb [20], z-test is inappropriate for small samples. The OPUS-IR authors propose the use of the standard t-test to decide on rules significance since the t-test tends to the z-test as the number of degrees of freedom increases. However, both z-test and t-test assume normality which in pratice cannot be guaranted. In this sense, using the KS approach is an advantage since no further distribution assumptions need to be considered.

Aumann and Lindell [2] propose an elaborated mechanism to identify and filter all the significante basic rules and sub-rules. They propose a notion of basic rule and an algorithm to find all significante "sub-rules" and "super-rules". The algorithm works as a post-mining step and builds a lattice of frequent sets to identify when a rule is basic or a sub-rule. The notion of super-rule is related to our notion of improvement. We also filter super-rules that do not bring about an improvement in the property of interest (in our case the KS-interest). However, applying improvement filtering does not require any sofisticated algorithm with lattice traversel. In fact, improvement filtering is computed on the fly, along rule derivation.

The QAR authors also present a mechanism to derive rules with more than one p.o.i. in the consequent. In pratice, it seems interesting to analyse several numerical properties in parallel. QAR has this feature as a post-processing step. We include this feature in the CAREN-DR engine during rule derivation. Thus, it is only required to specify the different properties to derive rules for.

Association rules have been used in subgroup discovery. APRIORI-SD [11] uses association rules to discover interesting subgroups with categorical properties of interest. Our approach enables the discovery of subgroups with numeric and categoric properties of interest. In this paper we employ the KS test to handle numeric properties.

8 Conclusion

We have introduced the concept of Distribution Rules as a generalization of association rules. We provide the basic concepts, such as the general form, support and objective interest of distribution rules. We also describe how to visualize distribution rules. DRs are particulary interesting when there is a numerical property of interest, although the concept can be extended to categorical properties as well. With classical association rules we would have to pre-discretize the numerical attribute of interest. With quantitative association rules, we would reduce the set of values in the consequent to a summary given by the mean or median. In the case of distribution rules, we keep the whole set of values of the property of interest and use these in graphical representations or post processing. Distribution rules can be presented as text or graphically and can be used in tasks of descriptive and predictive knowledge discovery.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
2. Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. *Journal of Intelligent Information Systems*, 2003.
3. P. J. Azevedo and A. M. Jorge. The class project. http://www.niaad.liacc.up.pt/~amjorge/Projectos/Class/, 2006.
4. R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. In *ICDE*, pages 188–197. IEEE Computer Society, 1999.
5. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In J. Peckham, editor, *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 255–264, Tucson, Arizona, 13–15 June 1997.
6. M. Carney, P. Cunningham, J. Dowling, and C. Lee. Predicting probability distributions for surf height using an ensemble of mixture density networks. In *Proceedings of the 22 International Conference on Machine Learning, ICML' 05, Bonn, Germany*, 2005.

7. W. J. Conover. *Practical Nonparametric Statistics - Third Edition*. John Wiley & Sons, New York, 1999.
8. U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.
9. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 13–23, New York, NY, USA, 1996. ACM Press.
10. E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Research Issues on Data Mining and Knowledge Discovery*, 1997.
11. B. Kavsek, N. Lavrac, and V. Jovanoski. Apriori-sd: Adapting association rule learning to subgroup discovery. In M. R. Berthold, editor, *Advances in Intelligent Data Analysis V*, volume 2810 of *Lecture Notes in Computer Science*, pages 230 – 241, Berlin Heidelberg, 2003. Springer-Verlag.
12. M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 273–282, New York, NY, USA, 1994. ACM Press.
13. W. Klősgen. Explora: A multipattern and multistrategy discovery assistant. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.
14. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *KDD '98: Proceedings of the fourth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 80–86, New York, NY, USA, 1998. ACM Press.
15. B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 125–134, New York, NY, USA, 1999. ACM Press.
16. C. J. Merz and P. Murphy. Uci repository of machine learning database. http://www.cs.uci.edu/∼mlearn, 1996.
17. A. Ozgur, P.-N. Tan, and V. Kumar. Rba: An integrated framework for regression based on association rules. In M. W. Berry, U. Dayal, C. Kamath, and D. B. Skillicorn, editors, *SDM '04: Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA*, 2004.
18. A. Silberschatz and A. Tuzhilin. On subjective measure of interestingness in knowledge discovery. In *KDD '95: Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 275–281. AAAi Press, 1995.
19. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 1996. ACM Press.
20. G. I. Webb. Discovering associations with numeric variables. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–388, New York, NY, USA, 2001. ACM Press.
21. H. Zhang, B. Padmanabhan, and A. Tuzhilin. On the discovery of significant statistical quantitative rules. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383, New York, NY, USA, 2004. ACM Press.

Tractable Models for Information Diffusion in Social Networks

Masahiro Kimura[1] and Kazumi Saito[2]

[1] Department of Electronics and Informatics, Ryukoku University
Otsu, Shiga 520-2194, Japan
[2] NTT Communication Science Laboratories, NTT Corporation
Seika-cho, Kyoto 619-0237, Japan

Abstract. When we consider the problem of finding influential nodes for information diffusion in a large-scale social network based on the *Independent Cascade Model (ICM)*, we need to compute the expected number of nodes influenced by a given set of nodes. However, a good estimate of this quantity needs a large amount of computation in the ICM. In this paper, we propose two natural special cases of the ICM such that a good estimate of this quantity can be efficiently computed. Using real large-scale social networks, we experimentally demonstrate that for extracting influential nodes, the proposed models can provide novel ranking methods that are different from the ICM, typical methods of social network analysis, and "PageRank" method. Moreover, we experimentally demonstrate that when the propagation probabilities through links are small, they can give good approximations to the ICM for finding sets of influential nodes.

1 Introduction

Recently, considerable attention has been devoted to investigating social networks [9,5,4,7,11,6], since the progress of the Internet, the World Wide Web, and blogs has enabled us to collect real large-scale social networks. Here, a social network is a network of relationships and interactions among social entities such as individuals, organizations and groups. Examples include email networks, hyperlink networks of web sites, trackback networks of blogs, and scientific collaboration networks. Since information, ideas, and influence can propagate through a social network in the form of "word-of-mouth" communications, it is an important research issue to find influential nodes for information diffusion in the underlying network in terms of sociology and marketing. Namely, it is significant to investigate the problem of finding nodes that generate a large spread of information. For example, Domingos and Richardson [2,12], and Kempe *et al.* [5] in particluar studied the *influence maximization problem*, that is, the problem of choosing a set of k nodes to target for initial activation such that it yields the largest expected spread of information, where k is a given integer.

In order to investigate these problems, we need a model of information diffusion in a social network. Although models for diffusion processes in a network

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 259–271, 2006.
© Springer-Verlag Berlin Heidelberg 2006

have been studied in various fields including epidemiology, sociology, marketing and physics [5,4], one of the conceptually simplest models is the *Independent Cascade Model (ICM)* used by Goldenberg *et al.* [3], Kempe *et al.* [5], and Gruhl *et al.* [4]. The ICM is a stochastic process model in which information propagates from a node to its neighboring nodes at each time-step according to some probabilistic rule. Therefore, when we consider the problem of finding sets of influential nodes in a social network based on the ICM, we need to compute the expected number $\sigma(A)$ of nodes influenced by a given set A of nodes. It is an open question to compute $\sigma(A)$ exactly by an efficient method, and so good estimates were obtained by simulating the random process many times [5]. However, such computations become very heavy for a large-scale social network.

In this paper, as natural special cases of the ICM, we propose two novel information diffusion models such that a good estimate of $\sigma(A)$ can be efficiently computed. Using large real data from a blog network and a scientific collaboration network, we experimentally explore properties of the proposed models. First, we experimentally compare the proposed models with the ICM, typical methods of social network analysis [13], and "PageRank" method [1] in terms of ranking methods to extract influential nodes, and show that the proposed models provide novel scalable ranking methods that can in general extract nontrivial nodes as influential nodes. We also demonstrate that when the propagation probabilities through links are small, the proposed models can provide good approximations to the ICM for finding sets of influential nodes in a social network. On the other hand, if we consider the influence maximization problem in the ICM, a provable performance guarantee for a natural greedy alogrithm was obtained by Kempe *et al.* [5]. We extend this result to the proposed models.

2 Independent Cascade Model

Based on the work of Kempe *et al.* [5], we recall the definition of the ICM, and an approximation theory for the influence maximization problem in the ICM.

2.1 Definition

We consider the ICM for the spread of a certain information through a social network represented by a directed graph. First, we call nodes *active* if they have accepted the information. We assume that nodes can switch from being inactive to being active, but cannot switch from being active to being inactive. When node u first becomes active at step t, it is given a single chance to activate each currently inactive *child* v, and succeeds with probablity $p_{u,v}$. Here, $p_{u,v}$ is a constant that is independent of the history of the process, and node v is called a *child* of node u and node u is called a *parent* of node v if there is a directed link (u, v) from u to v. If u succeeds, then v will become active at step $t + 1$. If multiple parents of v first become active at step t, then their activation attempts are sequenced in an arbitrary order, but performed at step t. Whether or not u succeeds, it cannot make any further attempts to activate v in subsequent rounds. The process terminates if no more activations are possible.

For an initial active set A, let $\sigma(A)$ denote the expected number of active nodes at the end of the process. We call $\sigma(A)$ the *influence* of target set A.

2.2 Approximation Theory

We consider the influence maximization problem in the ICM. Namely, for a given positive integer k, we consider finding a set A_k^* of k nodes to target for initial activation such that $\sigma(A_k^*) \geq \sigma(B)$ for any set B of k nodes based on the ICM. For this problem, we analyze the following natural greedy algorithm:

1. Start with $B = \emptyset$.
2. **for** $i = 1$ to k **do**
3. Choose a node v_i maximizing $\sigma(B \cup \{v_i\}) - \sigma(B)$.
4. Set $B \leftarrow B \cup \{v_i\}$.
5. **end for**

Let B_k denote the set of k nodes obtained by this algorithm. Then, Kempe *et al.* [5] proved that $\sigma(B_k) \geq (1 - 1/e)\ \sigma(A_k^*)$, that is, they presented an approximation guarantee for this algorithm. Their proof relies on the theory of *submodular functions* [8]. Here, for a function f that maps a subset of a finite ground set U to a nonnegative real number, f is called *submodular* if $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$ for any $u \in U$ and any pair $\{S, T\}$ of subsets of U with $S \subset T$. They proved the result of the approximation guarantee by showing that the function σ is submodular for the ICM.

However, for a naive implementation of this greedy algorithm, we need to compute the influence $\sigma(A)$ for each target set A. Since it is not clear how to evaluate $\sigma(A)$ exactly by an efficient method, Kempe *et al.* [5] obtained a good estimate by simulating the random process $10,000$ times for each target set. They argued that the quality of the approximation after $10,000$ iterations is comparable to that after $300,000$ or more iterations.

3 Proposed Models

We propose two novel information diffusion models as natural special cases of the ICM, and describe an approximate computation of influence $\sigma(A)$ for them. Moreover, we extend the approximation theory for the influence maximization problem by Kempe *et al.* [5] to the proposed models.

3.1 Definitions

We define two natural special cases of the ICM. Let A be the initial active set in the network, that is, the set of nodes that first become active at step 0. For nodes u and v in the network, let $d(u, v)$ denote the graph distance from u to v, and let $d(A, v)$ denote the graph distance from A to v, that is, $d(A, v) = \min_{u \in A} d(u, v)$. When there is no path from u to v, we set $d(u, v) = \infty$. Note that the value of $d(A, v)$ can be efficiently computed by graph theory [9].

First, we define the *Shortest-Path Model (SPM)*. The SPM is a special case of the ICM such that each node v has the chance to become active only at step $t = d(A, v)$. In other words, each node is activated only through the shortest paths from the initial active set. Namely, the SPM is a special type of the ICM where only the most efficient information spread can occur.

Next, we slightly generalize the SPM within the ICM, and define the *SP1 Model (SP1M)*. In the SP1M, each node v has the chance to become active only at steps $t = d(A, v)$ and $t = d(A, v) + 1$. In other words, node v cannot be activated excluding the paths from A to v whose length are equal to $d(A, v)$ or $d(A, v) + 1$.

We define the influence $\sigma(A)$ of target set A for the SPM and SP1M in the same way as the ICM.

3.2 Approximate Computation of Influence

We consider computing efficiently an approximate value of $\sigma(A)$ for the SPM and SP1M. Let V be the set of all the nodes in the network, N the number of elements of V, and V_A the set of nodes v such that $d(A, v) < \infty$. For any $v \in V$, let $P_t(v; A)$ denote the probability that v first becomes active at step t, and let $PA(v)$ denote the set of all the parent nodes of v. Here, note that $P_t(v; A) = 0$ for any $t \geq 0$ if $v \notin V_A$. Note also that for each $v \in A$, $P_t(v; A) = 1$ if $t = 0$, and $P_t(v; A) = 0$ if $t > 0$.

We begin with the SPM. We consider calculating $\sigma(A)$ from a computation of $P_t(v; A)$ for any $t \geq 0$ and $v \in V$. Note first that for any $v \in V_A$, $P_t(v; A) = 0$ if $t \neq d(A, v)$. Thus, we focus on $t = d(A, v)$ for any $v \in V_A$. Then, it is easily shown that $P_t(v; A)$ is computed by

$$P_t(v; A) = \sum_{W \subset PA(v)} P_{t-1}(W|PA(v); A)\, P_t(W \to v), \tag{1}$$

where the summation is taken over all subsets of $PA(v)$, $P_{t-1}(W|PA(v); A)$ denotes the probability that subset W first becomes active at step $t - 1$ in $PA(v)$, and $P_t(W \to v)$ denotes the probability that v is activated from W at step t when W is infectious. Here, we put $PA(v) = \{u_1, \cdots, u_K\}$, and use the following one-to-one correspondence between a subset W of $PA(v)$ and a binary K-vector $\boldsymbol{h} = (h_1, \cdots, h_K)$; for each k, $h_k = 1$ if $u_k \in W$, and $h_k = 0$ if $u_k \notin W$. Following Domingos and Richardson [2], we approximate the joint probabilities $\{P_{t-1}(W|PA(v); A);\, W \subset PA(v)\}$ by their maximal entropy estimates given the mariginals $\{P_{t-1}(u_k; A);\, k = 1, \cdots, K\}$. This yields

$$P_{t-1}(W|PA(v); A) = \prod_{k=1}^{K} P_{t-1}(u_k; A)^{h_k} \left(1 - P_{t-1}(u_k; A)\right)^{1-h_k}. \tag{2}$$

Note here that we can also obtain the same result by assuming that events $E_{1,t-1}, \cdots, E_{K,t-1}$ are independent, where each $E_{k,t-1}$ is the event that node u_k first becomes active at step $t - 1$. Thus, by (1) and (2), we have

$$P_t(v;A) = \sum_{\boldsymbol{h}} \left[\left\{ \prod_{k=1}^{K} P_{t-1}(u_k;A)^{h_k} \left(1 - P_{t-1}(u_k;A)\right)^{1-h_k} \right\} \left\{ 1 - \prod_{k=1}^{K} \left(1 - p_{u_k,v}\right)^{h_k} \right\} \right],$$

$$\begin{aligned} P_t(v;A) &= \sum_{\boldsymbol{h}} \prod_{k=1}^{K} P_{t-1}(u_k;A)^{h_k} (1 - P_{t-1}(u_k;A))^{1-h_k} \\ &\quad - \sum_{\boldsymbol{h}} \prod_{k=1}^{K} \left\{ P_{t-1}(u_k;A)\ (1 - p_{u_k,v}) \right\}^{h_k}\ (1 - P_{t-1}(u_k;A))^{1-h_k} \\ &= \prod_{k=1}^{K} \{ P_{t-1}(u_k;A) + (1 - P_{t-1}(u_k;A)) \} \\ &\quad - \prod_{k=1}^{K} \left\{ P_{t-1}(u_k;A)\ (1 - p_{u_k,v})\ +\ (1 - P_{t-1}(u_k;A)) \right\} \\ &= 1 - \prod_{k=1}^{K} \left(1\ -\ p_{u_k,v}\, P_{t-1}(u_k;A)\right). \end{aligned}$$

Under this approximation, we estimate $\sigma(A)$ as $\sigma(A) = \sum_{v \in V_A} P_{d(A,v)}(v;A)$.

Next, we consider the SP1M. In this case, for any $v \in V_A$, $P_t(v;A) = 0$ if $t \neq d(A,v)$, $d(A,v) + 1$. Thus, we focus on $t = d(A,v)$ and $t = d(A,v) + 1$ for any $v \in V_A$. In the same way as the case of the SPM, we approximate $P_t(v;A)$ by

$$P_t(v;A) = (1 - P_{t-1}(v;A)) \left\{ 1 - \prod_{u \in PA(v)} \left(1\ -\ p_{u,v}\, P_{t-1}(u;A)\right) \right\}.$$

Under this approximation, we estimate $\sigma(A)$ as $\sigma(A) = \sum_{v \in V_A} (P_{d(A,v)}(v;A) + P_{d(A,v)+1}(v;A))$.

As investigated by Leskovec *et al.* [6], it seems that large cascades of information diffusion happen rarely. We believe that this kind of real situations can be reasonably simulated by using SPM or SP1M with relatively small $p_{u,v}$. Using real social networks, we experimentally confirmed that the proposed estimation methods can be effective for the SPM and SP1M especially with relatively small $p_{u,v}$ (see Appendix). These results imply that for the SPM and SP1M, $\sigma(A)$ can be efficiently estimated in a reasonable situation.

3.3 Extension of Approximation Theory

For the SPM and SP1M, we consider the influence maximization problem, and investigate an approximation guarantee for the greedy algorithm defined in Sect. 2.2. We fix an integer k ($1 \leq k < N$). Let A_k^* be a set that maximizes the value of σ over all k-element subsets of V, and let B_k be the k-element set obtained by the greedy algorithm. Then, we can obtain the same result as that for the ICM.

Theorem 1. *In the SPM and SP1M, we have the following approximation guarantee for the greedy algorithm:* $\sigma(B_k) \geq (1 - 1/e)\sigma(A_k^*)$.

Proof. We prove this inequality in the same way as [5]. By the theory of submodular functions (see Theoerm 2.1 in [5]), it is sufficient to prove that σ is submodular in the SPM and SP1M. According to the proof of Theorem 2.2 in [5], we view the ICM in terms of *live* and *blocked* links. We first consider the SPM. Let X denote one sample set of outcomes for all the coin flips on the directed links in the network. Let $P(X)$ denote the probability of sample X. For any $u \in V$, let $S(u)$ be the set of shortest paths from u to each node in V, and let $L(u; X)$ be the set of live link paths from u with respect to X. We define $R(u; X)$ to be the set of nodes that belong to the paths in $S(u) \cap L(u; X)$. For any $A \subset V$, we define $\sigma_X(A)$ to be the number of nodes in $\cup_{u \in A} R(u; X)$. Then, we have $\sigma(A) = \sum_X P(X)\sigma_X(A)$, where the summation is taken over all samples. We can easily prove that σ_X is submodular and a nonnegative linear combination of submodular functions is also submodular. Hence, σ is submodular in the SPM. Similarly, we can also prove that σ is submodular in the SP1M.

4 Experimental Evaluation

Using real large-scale social networks, we experimentally explore properties of the proposed models.

4.1 Data Sets

We used two different data sets of large-scale social networks. The details of these data sets are given below.

Blog Network Data. First, we used a tackback network of blogs as an example of a social network. By tracing ten steps ahead the trackbacks from the blog of the theme "JR Fukuchiyama Line Derailment Collision" in the site "Theme salon of blogs" (`http://blog.goo.ne.jp/usertheme/`), we collected a large connected trackback network in May, 2005. Here, the total numbers of blogs and trackbacks were $12,047$ and $39,960$, respectively. Since bloggers discuss various topics and establish mutual communications by putting trackbacks on each other's blogs, we regarded a link created by a trackback as a bidirectional link for simplicity. We call this data set the BN data.

Collaboration Network Data. Next, we employ a collaboration network obtained from co-authorships of physics papers as an example of a social network, where each undirected link is regarded as a bidirectional link. We used the co-authorship network of the Los Alamos Condensed Matter e-print archives investigated by Palla *et al.* [11]. Here, the total numbers of nodes and undirected links were $30,561$ and $125,959$, respectively. The network consisted of 668 connected components, and the total number of nodes in the maximal connected component was $28,502$. We call this data set the CN data.

4.2 Experimental Settings and Fundamental Statistics

In our experiments, we assigned a uniform probability of p to each directd link in the network for the ICM, SPM, and SP1M, that is, $p_{u,v} = p$ for any directed link (u, v). As regards large and small propagation probabilities, we investigated $p = 10\%$ and $p = 1\%$, respectively.

According to the work of Kempe *et al.* [5], we estimated the influence $\sigma(A)$ of target set A in the ICM as follows: We started the process by initially activating A, and counted the number of active nodes at the end of the process. We then used the empirical mean obtained by simulating the stochastic process $10,000$ times as the estimate. However, these estimates needed very heavy computations. For example, for the CN data, the estimates of $\{\sigma(v);\ v \in V\}$ for the ICM with $p = 1\%$ needed about 3 hours, and those with $p = 10\%$ needed about 115 hours. Incidentally, in both cases of $p = 1\%$ and $p = 10\%$, it took about 5 and 20 minutes, respectively, to compute the estimates of $\{\sigma(v);\ v \in V\}$ for the SPM and SP1M based on the proposed estimation methods. Here, all our experimentation was undertaken on a single Dell PC with an Intel 3GHz Pentium D processor, with 2GB of memory. From these facts, we also confirm that as p increases, the processing time for estimating $\{\sigma(v);\ v \in V\}$ for the ICM much increases, while the processing times for the SPM and SP1M hardly change. Therefore, we can deduce that unlike the proposed models, the ICM needs a very large amount of computation for solving the influence maximization problem with $p = 10\%$ in a large-scale network based on the natural greedy algorithm. In particular, sophisticated techniques such as parallel computing must be needed to practically solve this problem for the ICM with $p = 10\%$ in our data sets.

When we estimated $\{\sigma(v);\ v \in V\}$ through $10,000$ simulations for the ICM, we also computed the standard deviation for each node. For example, for $p = 10\%$, the average standard deviations in the BN data and the CN data were 139.12 and $2,092.60$, respectively. Here, the average of $\{\sigma(v);\ v \in V\}$ in the BN data and that in the CN data were 87.80 and $1,586.59$, respectively. We see from these facts that the number of finally influenced nodes can greatly vary every simulation in the ICM.

From the above observations we deduce that a large amount of computation can be generally needed to obtain good estimates of $\{\sigma(v);\ v \in V\}$ for the ICM in a large-scale network, and so the ICM can be a computationally expensive model. Thus, for reference purposes, we also investigated the special case where the influence $\sigma(A)$ of target set A is estimated through 100 simulations in the ICM. We refer to this special model as the *ICM100*.

4.3 Ranking Problem

First, we consider extracting influential nodes from the network by ranking nodes based on influence measure. The ICM, ICM100, SPM, and SP1M can measure the influence of node v by $\sigma(v)$. On the other hand, "degree centrality", "closeness centrality", and "betweenness centrality" are commonly used as influence measure in sociology [13], where the degree of node v is defined as the number

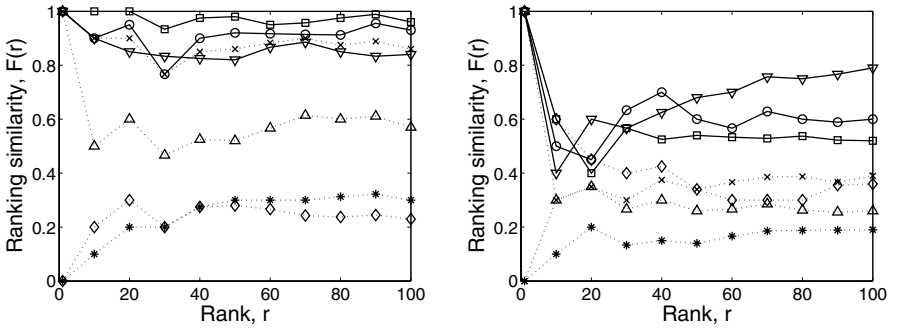


Fig. 1. Results for ranking similarities in the BN data. Left: $p = 1\%$. Right: $p = 10\%$. ("$\triangledown$": ICM100. "$\circ$": SPM. "$\square$": SP1M. "$\times$": degree centrality. "$\diamondsuit$": closseness centrality. "$*$": betweenness centrality. "$\triangle$": PageRank.)

of links attached to v, the closeness of node v is defined as the reciprocal of the average distance between v and other nodes in the network, and the betweenness of node v is defined as the total number of shortest paths between pairs of nodes that pass through v. We also consider measuring the influence of each node by its "authoritativeness" obtained by the "PageRank" method [1], since this is a well known method for identifying authoritative or influential pages in a hyperlink network of web pages. This method has a parameter ε; when we view it as a model of a random web surfer, ε corresponds to the probability with which a surfer jumps to a page picked uniformly at random [10]. In our experiments, we used a typical setting of $\varepsilon = 0.2$.

In terms of ranking methods for extracting influential nodes from the network, we compare the proposed models with the others for each value of p, so we introduce the *ranking similarity* $F(r)$ at rank r that quantifies the degree of similarity between two ranking methods. Based on F-measure, $F(r)$ is defined as follows: Let $L(r)$ and $L'(r)$ be the respective sets of top r nodes for the two ranking methods that we compare. Then, $F(r) = |L(r) \cap L'(r)| \ / \ r$, where $|S|$ indicates the number of elements in a set S. We focus on ranking similarities at high ranks since we are interested in extracting influential nodes.

Figs. 1 and 2 show the experimental results, where the ranking similarities $F(r)$ between the ICM method and the other methods are displayed at rank r $(1 \leq r \leq 100)$. Here, downward-pointing triangles, circles, squares, crosses, diamonds, asterisks, and upward-pointing triangles indicate the results for the ICM100, SPM, SP1M, degree centrality, closeness centrality, betweenness centrality, PageRank, respectively. We can observe that as ranking methods to extract influential nodes, the proposed models in general yield ranking results that are different from the ICM, typical methods of social network analysis, and PageRank method. This implies that the SPM and SP1M can provide novel ranking methods that in general extract nontrivial nodes as influential nodes. We can also observe that for $p = 1\%$, the ranking similarities of the proposed

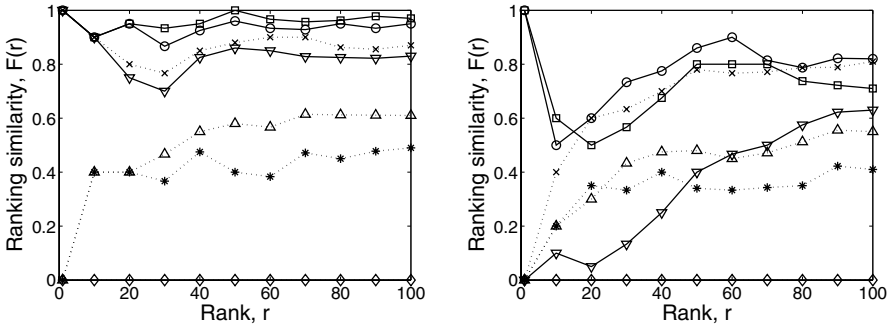


Fig. 2. Results for ranking similarities in the CN data. Left: $p = 1\%$. Right: $p = 10\%$. ("∇": ICM100. "$\circ$": SPM. "$\square$": SP1M. "$\times$": degree centrality. "$\Diamond$": closseness centrality. "$*$": betweenness centrality. "$\triangle$": PageRank.)

models and ICM were very high, and higher than those of the ICM100 and ICM. These results imply the following remarkable result: When p is small, the SPM and SP1M can give good approximations to the ICM in terms of ranking methods for extracting influential nodes in a social network. However, we note that the SPM and SP1M do not necessarily provide good estimates of $\{\sigma(v); v \in V\}$ for the ICM even if p is small. On the other hand, we can see that the ranking similarities of the ICM100 and ICM were not high for $p = 10\%$ in particular. These results imply that good estimates of $\{\sigma(v); v \in V\}$ for the ICM cannot necessarily be obtained by using 100 simulations.

Moreover, we note that unlike the ICM, the proposed models can provide scalable ranking methods such as the typical methods of social network analysis. Namely, the ranking methods based on the proposed models can practically be applied even to a large-scale social network with $p = 10\%$. In fact, their computational complexities are almost comparable to those of the betweenness centrality and closeness centrality methods. We believe that this property is important for a practical ranking method based on information diffusion in a social network.

4.4 Influence Maximization Problem

We further investigate whether or not the proposed models can approximate the ICM for extracting sets of influential nodes in a social network, when propagation probabilities through links are small. For this purpose, we employ the task of approximately solving the influence maximization problem in the ICM with $p = 1\%$. To perform this task, we apply the ICM, ICM100, SPM, and SP1M with $p = 1\%$ in the following way: As an approximate solution for a target set size k, we use the optimal k-element set obtained by the natural greedy algorithm based on each model. Let B_k^0, B_k^1, B_k^2, and B_k^3 denote the optimal k-element sets based on the ICM, ICM100, SPM, and SP1M, respectively. To simplify our explanation, let $\sigma^0(A)$ denote the influence $\sigma(A)$ of targe set A for the ICM with $p = 1\%$. We evaluate the

Table 1. Performance of approximate solutions for the influence maximization problem in the ICM with $p = 1\%$ in the BN data

Target set size	ICM	ICM100	SPM	SP1M
$k = 1$	3.87	3.87	3.87	3.87
$k = 10$	30.06	27.67	30.07	30.06
$k = 20$	51.84	44.40	51.84	51.87
$k = 30$	71.83	57.79	71.96	72.01

Table 2. Performance of approximate solutions for the influence maximization problem in the ICM with $p = 1\%$ in the CN data

Target set size	ICM	ICM100	SPM	SP1M
$k = 1$	3.78	3.78	3.78	3.78
$k = 10$	33.35	30.61	33.44	33.42
$k = 20$	59.40	51.80	59.39	59.53
$k = 25$	71.59	59.76	71.33	71.69

performance of an approximate solution B_k^i by the value of $\sigma^0(B_k^i)$. Of course, we estimated $\sigma^0(B_k^i)$ through $10,000$ simulations.

In our experiments, we examined such approximate solutions from $k = 1$ to $k = 30$ in the BN data, and from $k = 1$ to $k = 25$ in the CN data. Tables 1 and 2 show the experimental results, where the value of $\sigma^0(B_k^i)$ $(i = 0, 1, 2, 3)$ for each target set size k is displayed. We observe that the evaluation values for the proposed models were almost the same as those for the ICM, and better than those for the ICM100. These results imply that when p is small, the proposed models can provide good approximations to the ICM for finding sets of influential nodes in a social network.

We also examined the processing times for computing the approximate solutions. Let $t_+(k)$ be the processing time for computing B_k^i given B_{k-1}^i. Fig. 3 shows the processing time $t_+(k)$ at target set size k for each model, where left-pointing triangles, downward-pointing triangles, circles, and squares indicate the ICM, ICM100, SPM, and SP1M, respectively. We can see that as k increases, $t_+(k)$ does not increase so much for the SPM and SP1M, but it substantially increases for the ICM and ICM100. This implies that the methods based on the proposed models can be practically performed even for a large target set size k. Namely, we can see that the proposed models are also scalable to solve the influence maximization problem based on the greedy algorithm. On the other hand, the total processing time for obtaining $\{B_k^i;\ k = 1, \cdots, 30\}$ in the BN data and that for obtaining $\{B_k^i; k = 1, \cdots, 25\}$ in the CN data were as follows: In the BN data, the total processing times for the ICM, ICM100, SPM, and SP1M were about 5 days, 1 hour, 19 minutes, and 1 hour,

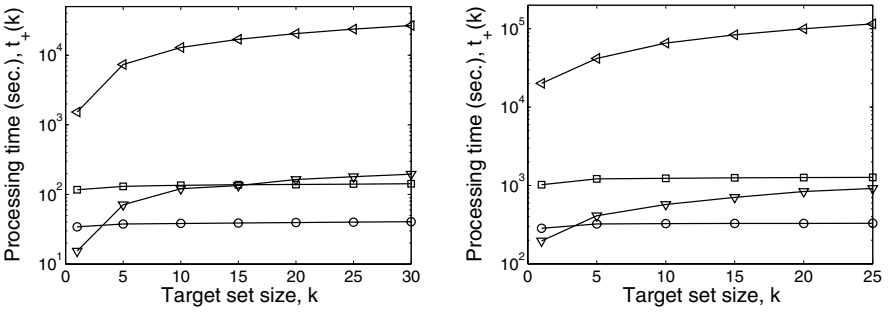


Fig. 3. Processing time $t_+(k)$ for target set size k. Left: BN data. Right: CN data. ("$\triangleleft$": ICM. "$\triangledown$": ICM100. "$\circ$": SPM. "$\square$": SP1M.)

respectively. In the CN data, the total processing times for the ICM, ICM100, SPM, and SP1M were about 21 days, 4 hours, 2 hours, and 8 hours, respectively. Note here that the total processing times for the ICM were more than 100 times those for the ICM100 since $\sigma^0(B_k^0) > \sigma^0(B_k^1)$ for $k > 1$. These results show that a very large amount of computation is needed to solve the influence maximization problem for the ICM in a large-scale social network by using the greedy algorithm. Moreover, the following interesting observation is made: For the influence maximization problem in the ICM with small p, the methods based on the proposed models can be much faster than the ICM-based method, and can provide as good approximate solutions as the ICM-based method.

5 Conclusions

We have proposed two natural models for information diffusion in a social network, called the SPM and SP1M, such that the influence $\sigma(A)$ of a target set A can be efficiently estimated in a reasonable situation. For the influence maximization problems in the proposed models, we have provided a provable performance guarantee for the natural greedy algorithm. Using real large-scale social networks, we have experimentally explored properties of the SPM and SP1M. First, we have demonstrated that the proposed models can provide novel scalable ranking methods for extracting influential nodes in a social network. Next, we have demonstrated that when the propagation probabilities through links are small, they can give good approximations to the ICM for finding sets of influential nodes in a social network. Moreover, we have demonstrated that for solving the influence maximization problem based on the greedy algorithm, the proposed models can be scalable, and also be much faster than the ICM. Hence, we consider that the SPM and SP1M can be important models for social network analysis based on information diffusion.

References

1. Brin, S., and Page, L., The anatomy of a large-scale hypertextual Web search engine, In *Proc. WWW'98* (1998), 107–117.
2. Domingos, P., and Richardson, M., Mining the network value of customers. In *Proc. KDD'01* (2001), 57–66
3. Goldenberg, K. J., Libai, B., and Muller, E., Talk of the network: A complex systems look at the underlying process of word-of-mouth, *Marketing Letters* **12** (2001), 211–223.
4. Gruhl, D., Guha, R., Liben-Nowell, D., and Tomkins, A., Information diffusion through blogspace, In *Proc. WWW'04* (2004), 491–501.
5. Kempe, D., Kleinberg, J., and Tardos, E., Maximizing the spread of influence through a social network, In *Proc. KDD'03* (2003), 137–146.
6. Leskovec, J., Singh, A., and Kleinberg, J., Patterns of influence in a recommendation network, In *Proc. PAKDD'06* (2006), 380–389.
7. McCallum, A., Corrada-Emmanuel, A., and Wang, X., Topic and role discovery in social networks, In *Proc. IJCAI'05* (2005), 786–791.
8. Nemhauser, G. L., and Wolsey, L. A., *Integer and Combinatorial Optimization*. Wiley, New York, 1988.
9. Newman, M. E. J., Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality, *Physical Review E* **64** (2001), 016132.
10. Ng, A. Y., Zheng, A. X., and Jordan, M. I., Link analysis, eigenvectors and stability, In *Proc. IJCAI'01* (2001), 903–901.
11. Palla, G., Derényi, I., Farkas I., and Vicsek, T., Uncovering the overlapping community structure of complex networks in nature and society, *Nature* **435** (2005), 814–818.
12. Richardson, M., and Domingos, P., Mining knowledge-sharing sites for viral marketing, In *Proc. KDD'02* (2002), 61–70.
13. Wasserman, S., and Faust, K., *Social Network Analysis*. Cambridge University Press, Cambridge, 1994.

Appendix

Performance Evaluation of Influence Estimation. In Sect. 3.2, we have proposed methods to estimate the influence $\sigma(A)$ of a target set A for the SPM and SP1M. Using several real social networks, we experimentally confirmed that the methods can be effective for the SPM and SP1M with relatively small propagation probabilities through links. Here, we describe the experimental results for the estimates of $\{\sigma(v);\, v \in V\}$ in the BN data.

In the experiments, we examined both cases of $p = 1\%$ and $p = 10\%$. First, we estimated the values of $\sigma(v)$ for the SPM and SP1M through simulating the stochastic processes $10,000$ times like the case of the ICM, and adopted them as the true values of $\sigma(v)$. Then, the average m and standard deviation s of $\{\sigma(v);\, v \in V\}$ were as follows:

SPM: ($p = 1\%$: $m = 1.081$, $s = 0.126$), ($p = 10\%$: $m = 4.212$, $s = 4.061$).
SP1M: ($p = 1\%$: $m = 1.085$, $s = 0.138$), ($p = 10\%$: $m = 8.322$, $s = 10.668$).

Let $\hat{\sigma}(v)$ denote the estimate of $\sigma(v)$ by the proposed methods for the SPM and SP1M. We measured the approximation performance by error $\mathcal{E} = \sum_{v \in V} |\sigma(v) - \hat{\sigma(v)}| \ / \ N$. The results were as follows:

SPM: $(p = 1\%\text{: } \mathcal{E} = 0.002)$, $(p = 10\%\text{: } \mathcal{E} = 0.045)$.
SP1M: $(p = 1\%\text{: } \mathcal{E} = 0.003)$, $(p = 10\%\text{: } \mathcal{E} = 0.479)$.

These results show that the proposed estimation methods can be effective in a reasonable situation.

Efficient Spatial Classification Using Decoupled Conditional Random Fields

Chi-Hoon Lee, Russell Greiner, and Osmar Zaïane

Department of Computing Science, University of Alberta, Edmonton, AB, Canada

Abstract. We present a discriminative method to classify data that have interdependencies in 2-D lattice. Although both Markov Random Fields (MRFs) and Conditional Random Fields (CRFs) are well-known methods for modeling such dependencies, they are often ineffective and inefficient, respectively. This is because many of the simplifying assumptions that underlie the MRF's efficiency compromise its accuracy. As CRFs are discriminative, they are typically more accurate than the generative MRFs. This also means their learning process is more expensive. This paper addresses this situation by defining and using "Decoupled Conditional Random Fields (DCRFs)", a variant of CRFs whose learning process is more efficient as it decouples the tasks of learning potentials. Although our model is only guaranteed to approximate a CRF, our empirical results on synthetic/real datasets show that DCRF is essentially as accurate as other CRF variants, but is many times faster to train.

1 Introduction

Much of data mining deals with ways to learn classifiers from data samples. While many standard learning systems (*e.g.*, SVM, Logistic Regression, Naïve Bayes, Decision Trees, etc.) are designed to deal with independent and identically distributed data, this paper deals with interdependent data — viz., classifying regions in a 2-D lattice. In particular, we consider the task of detecting and delimiting tumors in Magnetic Resonance (MR) images of a patient's brain, which involves labeling each pixel as either tumor or non-tumor. Since most tumors are contiguous regions, we expect the labels of spatially adjacent pixels to belong to the same class, assuming they have sufficiently similar features.

Many effective region classifiers incorporate spatial constraints to encode the fact that the labels of neighboring pixels are typically correlated. In particular, there are a number of "random field" approaches for such tasks, including generative models like Markov Random Field (MRF) [12,8], as well as discriminative models, including Conditional Random Field (CRF) [10] and its variants — Discriminative Random Field (DRF) [9], Associative Markov Nets (AMN) [18] , and our recent Support Vector Random Field (SVRF) [11]. As an MRF assumes conditional independence among observations given class labels, their learning procedures tend to be faster than the discriminative models (variants of CRFs); however, this assumption means they are typically not as accurate. The more accurate models, unfortunately, can be prohibitively slow to train, which may not be tolerable to a data mining task. We therefore propose a novel variant to our discriminative random fields model to make them more efficient

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 272–283, 2006.
© Springer-Verlag Berlin Heidelberg 2006

to train: we develop a "decoupled" learner, DCRF that reduces the expense of learning the random fields. We found that, as expected, the resulting DCRF is much faster to train than other CRF variants. Moreover, we were pleasantly surprised to find that this improvement in speed did not cost a degradation in accuracy!

Section 2 presents a quick overview of related systems. It motivates our approach by noting that these related systems – especially the ones that produce accurate labelings – can be very slow to train. Section 3 introduces our novel "Decoupled Conditional Random Field" (DCRF) approach, and provides algorithms for both learning the parameters and for inference (*i.e.*, classification — here segmentation). Section 4 demonstrates the accuracy and efficiency of our model by presenting experimental results over various domains, including the challenging real-world problem of segmenting brain tumor from MRI scans.

2 Related Work

There are now many systems for learning the spatial correlations; this paper focuses on ones based on random fields.

An Markov Random Field (MRF)is a *generative* approach that models the joint probability distributions over a set of instances $\mathbf{x} = \langle x_i \rangle$ (where each x_i corresponds to a vector of values describing the i^{th} pixel) and their associated class labels $\mathbf{y} = \langle y_i \rangle$. As with other random fields, a MRF provides a form for computing $P(\mathbf{y} \mid \mathbf{x})$, based on both properties of each instance (*i.e.*, "pixel") as well as features of their "neighbors" (*i.e.*, "properties and perhaps labels of adjacent pixels"), towards returning the most likely $\mathbf{y}^*(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x})$.

In the MRF framework, the posterior over the n joint labels $\mathbf{y}$ given the observations $\mathbf{x}$ is $P(\mathbf{y} \mid \mathbf{x}) \propto P(\mathbf{y}) P(\mathbf{x} \mid \mathbf{y}) = P(\mathbf{y}) \prod_i^n P(x_i \mid y_i)$. Estimating the likelihood is computationally tractable as it is factored as $P(\mathbf{x} \mid \mathbf{y}) = \prod_i P(x_i \mid y_i)$. As this factorization is only a crude approximation to reality, this approach will typically produce inferior labels. The prior $P(\mathbf{y})$ can explicitly incorporate dependencies among the labels. Considering the equivalence between MRF and Gibbs Distributions [1], the posterior is formulated as

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left[\sum_{c \in C} V_c(\mathbf{y}_c) + \sum_{i \in S} \log(P(x_i \mid y_i))\right] , \tag{1}$$

where C is a set of cliques defined in 2-D lattice.$V_c(\mathbf{y})$ is a clique potential function of labels for clique $c \in C$, S is the set of nodes (*i.e.*, pixels), and the "partition function" $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \exp\left[\sum_{c \in C} V_c(\mathbf{y}'_c) + \sum_{i \in S} \log(P(x_i \mid y'_i))\right]$ is used to normalize the resulting values. Notice $V_c(\mathbf{y}_c)$ depends only on the labels $\{y_i\}$, but not on the information about the pixels $\{x_i\}$. Therefore, a MRF prefers a set of labels $\mathbf{y}^*$ where neighbors have the same value [1,12], independent of properties of these pixels. Also, as the partition function $Z(\mathbf{x})$ involves summing over all $|L|^n$ possible labelings (assuming there are $|L|$ labels for each pixel), it is very expensive to compute the exact value of the partition function.

A discriminative model, Conditional Random Field (CRF) [10], attempts to overcome the disadvantages of a MRF — notably its conditional independence assumption

and the absence of observation information in the second potential — by directly modeling the posterior distribution $P(\mathbf{y} \mid \mathbf{x})$ as

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i \in S}\left[\Phi_{\mathbf{w}}(y_i, \mathbf{x}) + \sum_{j \in N_i} \Psi_{\nu}(y_i, y_j, \mathbf{x})\right]\right) \quad (2)$$

which directly computes a posterior distribution without modeling the prior $P(\mathbf{y})$. The notation is essentially the same as in Equation 1: $Z(\mathbf{x})$ is the partition function, S is the set of pixels in an image, $\mathbf{x} = \langle x_i \rangle$ is the set of descriptions of those pixels, and $\mathbf{y} = \langle y_i \rangle$ is the set of labels. Here N_i is the set of neighbors of node x_i — in 2D, the pixel at location (a, b) has 4 neighbors, at $(a-1, b)$, $(a+1, b)$, $(a, b-1)$ and $(a, b+1)$ [1,8]. For notation, "$\Phi_{\mathbf{w}}(y_i, \mathbf{x})$" is called the "Association" potential, which deals with a single instance. While its value can depend on all of $\mathbf{x}$, it typically relies only on x_i. The "$\Psi_{\nu}(y_i, y_j, \mathbf{x})$" term is called the "Local-Consistency" (or "Interaction") potential in variants of CRF such as SVRFs and DRFs; it is typically used to prefer labeling that assign the same class labels to neighboring pixels. (We can view $\Psi_{\nu}(\cdot)$ as a data dependent smoothing function; this differs from a MRF, which instead use only a "data independent" term.) Here, $\mathbf{w}$ and ν refer to the parameters associated with these potential functions.

Note that a CRF and its variants — DRFs and SVRFs— typically produce better accuracy than their generative alternative MRF. However, their good performance comes at a cost: the learning process is significantly more expensive. For example, the learning task in DRF and SVRF involves estimating the parameters $\mathbf{w}$ and ν that maximize the (log)likelihood of the given data sample, and both systems use a regularization term to avoid overfitting. The log-likelihood is formulated as

$$\langle \hat{\mathbf{w}}, \hat{\nu} \rangle = \underset{\mathbf{w},\nu}{\operatorname{argmax}} \left\{ \sum_{k=1}^{M} \sum_{i=1}^{S} \Phi_w(y_i, \mathbf{x}) + \sum_{j \in N_i} \Psi_{\nu}(y_i^{(k)}, y_j^{(k)}, \mathbf{x}^{(k)}) - \log(Z^{(k)}(\mathbf{x})) \right\} - \frac{\nu^T \nu}{2\tau} \quad (3)$$

Although a SVRF can significantly improve the accuracy of a DRF, especially when features may be correlated, the study [11] has shown that selecting the appropriate τ in a SVRF and a DRF is a non-trivial task, which makes the learning procedures more challenging and costly. Associative Markov Nets (AMN) [18], which discriminatively train Markov nets, exploit the spatial correlations by adopting the maximum-margin principle of maximizing the margin between target labels and the best runner-up label assignments. Hence, this process employs the same ideas underlying SVMs. (Note that our SVRF differs by actually performing the same basic computations that an SVM performs.) A Boosted Random Field (BRF) [19] combines the set of iid classifiers that correspond to Association potentials. Each potential in a BRF is trained on a specific class to quantify the likelihood of a class on a pixel. Hence, BRF does not explicitly consider the spatial correlation.

We see there are problems in training each of the systems mentioned in this section: some are inaccurate (as they use inappropriate models), while others require too much computation time.

3 The DCRF System

This section presents the foundations to formalize our Decoupled Conditional Random Field, DCRFof random fields. We first motivate our approach of decoupling the training of the two potentials, then discuss inference — *i.e.*, how to use the resulting system to classify pixels in an image.

First, if we ignore the dependencies among the labels of the pixels (*i.e.*, assume that they are independent and identically distributed), we would use only the "Association" potential, which attempts to maximize

$$P_A(\,\mathbf{y}\,|\,\mathbf{x}\,) \quad \propto \quad \exp\left(\sum_{i\in S} \Phi(y_i, \mathbf{x})\right) \tag{4}$$

Many existing classifiers (*e.g.*, Naïve Bayes, Logistic Regressions, SVM, etc.) are (perhaps implicitly) attempting to optimize Equation 4.

Alternatively, a discriminative model that only considers spatial coherence would attempt to optimize

$$P_{LC}(\,\mathbf{y}\,|\,\mathbf{x}\,) \quad \propto \quad \exp\left(\sum_{i\in S} \Psi(y_i, y_{N_i}, \mathbf{x})\right) \tag{5}$$

where y_{N_i} are the labels of i's neighbors.

Equation 4 and 5 provide different frameworks for approximating the posterior probability distributions $P(\,\mathbf{y}\,|\,\mathbf{x}\,)$. Each is only partial, in that the first (second) does not properly incorporate spatial coherence (resp., the local observations).

Notice typical CRF models involve the sum of these equations — written in log space as

$$\sum_{i\in S} \Phi(y_i, \mathbf{x}) \quad + \quad \sum_{i\in S} \Psi(y_i, y_{N_i}, \mathbf{x}) \tag{6}$$

(Compare to Equation 2. Note that the neighborhood is considered in $\Psi(\cdot)$ explicitly.)

We now observe that each classifier form in Equation 6 follows MAP formulations for the joint probability over labels: that is, we can approximate the global optimal joint class labels by maximizing the local posterior probability distribution using the principles of pseudo-likelihood and Iterative Conditional Modes (ICM)[1] [3] — *i.e.*, $P(\,\mathbf{y}\,|\,\mathbf{x}\,) = \prod_{i\in S} P(\,y_i\,|\,y_{N_i}, \mathbf{x}\,)$. Thus, for each pixel i, the log of ensemblized posterior distribution $P(\,y_i\,|\,y_{N_i}, \mathbf{x}\,)$ given its neighbors y_{N_i} is:

$$\Phi_{\mathbf{w}}(y_i, \mathbf{x}) \; + \; \sum_{j\in N_i} \Psi_\nu(y_i, y_j, \mathbf{x}) \tag{7}$$

[1] Pseudo-likelihood and ICM are only guaranteed to achieve local maxima, the discussion of the global optimality issues is beyond the scope of this paper.

N.b., as we will only be seeking the argmax, we do not need to consider the normalizing "$-\log(z_i)$" term that shown in Equation 3, as it will be constant here.

Equation 7 shows that we can approximate a CRF model using a decoupled system, corresponding to the simple sum of two different potentials, *which are learned separately*. (This differs from standard ensemble methods [4], as we are directly combining *potentials* rather than classifiers.) However, there is one remaining question: how to deal with the relative scaling issues when combining of the two potentials. This will be discussed in the following sections. We will also see that, as expected, it is much faster to learn these individual summands *individually*, before combining them. Our empirical evidence shows that, surprisingly, the resulting DCRF system can be as accurate!

Association-only Potential. The association potential provides a local posterior for each pixel: $P_A(\, y_i \,|\, x_i \,)$. Our "decoupling" principle allows us to select a function that quantifies the conditional probability for a given observed instance. We incorporate a maximal margin approach where the two classes of pixels are classified based on a hyperplane that maximizing the distances between the two classes.

As suggested above, we will consider a potential based on SVMs; note this method inherits the SVM's relative insensitivity to class imbalance, and its ability to typically outperform other discriminative classifiers such as GLMs, especially in cases where the classes overlap [17], which is common case in imaging applications.

We select the hyperplane by solving the following optimization problem (over the t instances):

$$\begin{aligned} &\max_{w,b,\gamma} \quad \gamma \\ &\text{subject to} \quad y_i(w^T x_i + b) \geq \gamma, i = 1, \cdots, t \,; \;\; \|w\|^2 = 1 \end{aligned} \tag{8}$$

where γ is a margin, b is a bias term, and the vector w is normal to a hyperplane that we are seeking, which separates the positive from the negative examples. Using its dual formulation with dual variables α_i, we solve this optimization problem using Quadratic Programming (QP) over the α_is, to produce $f(x) = \sum_{i=1}^{t} \alpha_i \, y_i \, x_i^T \, x + b$ then use the decision function *sign* ($f(x)$) to classify the test instance x. Note this learning process requires only polynomial time. Our implementation actually uses Sequential Minimal Optimization (SMO) [14], which is an efficient implementation.

Notice that $f(x)$ computes the distance to the hyperplane from the instance x. We can use this to compute (something like) a posterior probability function [15,13]:[2]

$$\Phi_{\mathbf{w}}(y_i, \mathbf{x}) \;=\; \frac{1}{1 + \exp(A_A \times y_i(\mathbf{w}^T x_i) + B_A)} \tag{9}$$

using the parameters A_A and B_A that optimize the fit of the training data to a sigmoid function [15,11].

Local-Consistency-only Potential. We use our "local-consistency-only" potential to model the "neighborhood coherence" between pixels. Its goal is to encourage "similar"

[2] Of course, we augment the instance x_i by including a constant 1, and hence the $\mathbf{w}$ include a "constant" term as well.

instances within the specified each neighborhood to have the same labels. Although we can use the associated potential as a stand-alone decision function, its function here is mainly to smooth regions (and hence remove errors) produced by the Association-only potential.

For similar instances in a neighborhood to have similar (in our discrete case, "identical") class labels, we introduce a max-margin based potential, which tries to make the labels of a testing instance the same as the labels of its neighbors. This potential learns a pairwise max-margin model that quantifies the likelihood that two pixels will have the same class labels, given their descriptions:

$$\Psi_\nu(y_i, y_j, \mathbf{x}) \;=\; I(y_i, y_j) \times [\nu^T \langle\, \psi(x_i, x_j)\,\rangle] \tag{10}$$

where $I(y_i, y_j)$ returns $+1$ if $y_i = y_j$, and -1 otherwise. (We define $\psi(x_i, x_j)$ below.) Equation 10 reduces the pairwise discriminative learning problem to the binary class problem, over similar versus dissimilar classes. That is, we apply QP to the training set

$$S_{new} = \quad \{\; (\psi(x_r, x_j),\; I(y_r, y_j)) \mid j \in N_r \;\}$$

over all instances r with neighbors $j \in N_r$, to find the optimal parameter ν.

Note that each pair of pixels is projected by $\psi(\cdot)$ onto a similarity feature space. For instance, we could use $\psi(x_i, x_j) = x_i^T x_j$ that produces a scalar: the cosine measure of the similarity. Note this attains its largest value as the two vectors match one another. Due to "localized" neighborhood system we consider for Local-consistency potential, the increment only grows linearly with the number of pixels. Notice that feature-wise space depends on $\psi(\cdot)$.

As we will need to combine this potential with the Association-only one, we need to produce values within a "comparable" range. We therefore convert Equation 10 to the posterior probability scale, using the same transformation used to produce Equation 9.

$$\Psi(y_i, y_j, \mathbf{x}) = \frac{1}{1 + \exp\left(A_{LC} \times I(y_i, y_j)(\nu^T \langle\, \psi(x_i, x_j)\,\rangle) + B_{LC}\right)} \tag{11}$$

where again A_{LC} and B_{LC} are set to optimize the fit to a sigmoid, which produces a probability distribution as in Association-only potential

3.1 Inference

Our goal in producing this DCRF system is then to find relevant regions within images — *e.g.*, tumor regions within MR images of a brain. This involves inferring a binary label (tumor versus non-tumor) for each individual pixel. As noted above, this corresponds to computing the most likely vector $\mathbf{y}^* = \text{argmax}_{\mathbf{y}}\, P(\,\mathbf{y} \mid \mathbf{x}\,)$ given the evidence vector $\mathbf{x}$, based on the (possibly unnormalized) potential functions. In our case, we will use the potential functions in Equation 7, which is the sum of the Association-only $P_A(\,\cdot\,)$ (Equation 4) and Local-Consistency-Only $P_{LC}(\,\cdot\,)$ (Equation 5) potentials. While this exact computation can be expensive, there are several existing approximation algorithms for CRF, including Iterative Conditional Modes (ICM) [3], Graph-Cuts (GC) [2], and Loopy Belief Propagation (LBP) [6]. DCRF uses ICM since it converges

quickly and has been shown empirically to produce accurate results [11,1].[3] Also, while ICM may converge to local optima for the joint distribution problem, it works sufficiently well by iteratively propagating the belief for each pixel to its neighboring pixels:

$$y_i^* = \underset{y_i \in \{+1,\,-1\}}{\operatorname{argmax}} P(\, y_i \,|\, y_{N_i}, \mathbf{x}\,) = \underset{y_i \in \{+1,\,-1\}}{\operatorname{argmax}} \Phi(y_i, \mathbf{x}) \;+\; \sum_{j \in N_i} \Psi(y_i, y_j, \mathbf{x}) \quad (12)$$

Of course, we could add the normalization factor z_i in Equation 12, which constrains outputs to follow probability axioms. However, the constant factor is irrelevant, since our inference approach seeks only the most likely value.

3.2 Complexity

Our DCRF model uses Quadratic Programming to learn the parameters, within SMO. Assuming each image has S pixels, and each pixel has E neighbors then learning the Association-only potential requires $O(S^2)$ steps per image, and Local-Consistency-only potential requires $O(\,(S \times E)^2\,)$ per image. Note that in our paper, we used E is 4.

Inference (here, classifying the regions in a test image) requires $O(\,S + (S \times E)\,)$ per iteration. Empirically, we found that ICM converged after 5 iterations, on average.

4 Experiments

We implemented the Decoupled CRF described above, DCRF, and compared it with other random field techniques on both synthetic and real-world tasks. As many imaging tasks are very imbalanced (in that the "positive" class includes only a small percentage of the pixels), the standard evaluation criteria of "accuracy" is problematic. We therefore use the Jaccard score — $J = \frac{TP}{TP+FP+FN}$ — to measure its performance, using true positives (TP), false positives (FP), and false negatives (FN).

Synthetic image sets. The primary goal in using the synthetic data sets is to see how the various algorithms segment objects in the presence of noise. We therefore evaluated these techniques over 15 synthetic image sets, each with its own shape, whose intensities were each independently corrupted by noise generated from $\mathcal{N}(0, 1)$. Here each image is of size 64-by-64 (4096 pixels). Note that some of image sets are significantly imbalanced, while others are balanced.

Figure 1 shows some of the experiment results. All Jaccard scores and elapsed learning times that appear are averaged over 3-fold cross-validation. Each row in Figure 1 presents one example, showing (from left to right), the test images, the true labels, and outputs from Logistic Regression (LR), DRF, SVM, SVRF, and DCRF. We see that, overall, SVRF and DCRF are most accurate. Especially when the test images are imbalanced (the first row in Fig. 1), LR (third column) and DRF (fourth column) produce degraded outputs caused by the poor parameter estimations from the imbalanced data.

[3] While GC and LBP are often considered be the best inference methods, even if the graph structure has loops, we used ICM for the reasons shown above. Note this issue is orthogonal to the goal of this paper, which is to compare the training time and accuracy of our DCRF to other CRF-related models.

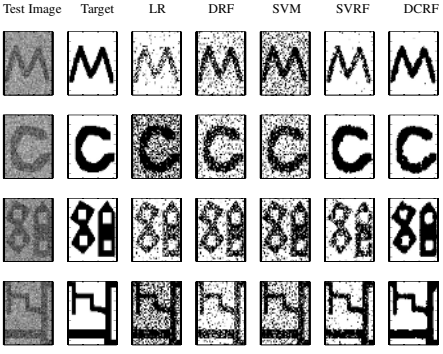


Fig. 1. Results from synthetic image sets. Rows 1 to 5 from the top down correspond respectively to datasets 7, 3, 10, and 11 in Fig. 2

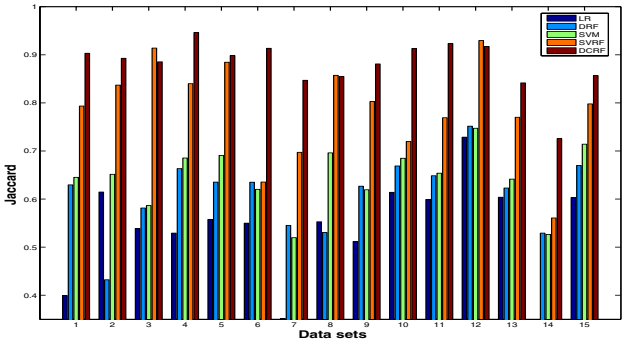


Fig. 2. Average Jaccard scores on 15 synthetic data sets

The second row illustrates the sensitivity of the regularization term τ in the SVRF frameworks. Although the correct value for this parameter can produce good segmentation results, in general it is not trivial to find such "good" values. While we can use cross-validation method to estimate this parameter, others [11,9] have shown that this does not guarantee acceptable performance. Also, note that SVM-based approaches appear robust to the class imbalance, as empirically shown in [11].

Figure 2 shows that DCRF and SVRF are the two best performers overall, at this segmentation task, dealing with both the balanced and imbalanced data: Each was significantly better than the others at the $p < 1.14\text{E-}12$ level based on a paired example t-test; moreover, DCRF performs better than SVRF at the $p < 0.0037$ level. Note that SVRF can sometimes produce better results than DCRF — see data sets 3, 9, and 12 in Figure 2. Here, we assume that SVRF found good estimates for τ. Data sets 6, 7, 9, 12, and 14 show that good estimation of the regularization in DRF performs better than SVM.

Table 1. Average elapsed learning time (seconds)

	DRF	SVRF	DCRF
Synthetic	1581.3	714.5	21.2
Brain Tumor	1392.4	1209.4	82.3

The first row of Table 1 reports the average learning time for DRF, SVRF and DCRF over these 15 cases. Notice first that our DCRF requires *significantly* less time than the other two approaches — 30 times faster than SVRF (and so $p < 1.165\text{E-}17$) and over 70 times faster than DRF. This is because there are fast ways to solve DCRF's underlying QPs which we attribute to the observation that the SVRF learner regards the Association potential as a constant while learning the Local consistency potential, but a DRF attempts to optimize both potentials simultaneously. (Note this is more than compensates for the fact that DRF's Logistic Regression learning, by itself, would be faster than SVRF's QP.) Finally, recall that DCRF does not compute the partition function during the training.

Real-world problems. We next applied these various learners to the task of segmenting brain tumors from MR images. Tumor segmentation is challenging for many reasons, including the differences between the brains of different individuals, and the fact that the same intensity values can be a tumor in one part of the image, but normal tissue in another [5,7]. Automatic tumor segmentation would be very useful, as it would enable radiation oncologists to effectively locate the tumor, with sufficient precision that they can use this to perform diagnosis and to plan treatments.

Our experimental data sets consists of 13 volumes taken from 7 patients, each having either a grade 2 astrocytoma, an anaplastic astrocytoma, or a glioblastoma multiform. We focused only on the axial MRI slices — there were around 21 such slices per patient-visit. For each slice, there are three complete images, corresponding to three standard modalities, called "t1", "t2" and "t1c" [7]. These represent challenging cases since the tumor area is typically heterogeneous.

We used the multi-scale feature set based on [16], which contains traditional image-based features in addition to three types of 'alignment-based' features: spatial probabilities for each of the 3 normal tissue types (white matter, gray matter, CSF), spatial expected intensity maps, and a characterization of left-to-right symmetry; each measured at multiple scales. As with many of the related works on brain tumor segmentation (such as [5,20]), our training is a patient-specific scenario, where training data for the classifier is obtained from the patient to be segmented. Note that pixels to be tested are from a brain slice that is different from the slice containing the training pixels.

In our experiment, we evaluated the following 7 classifiers on the 13 different time points from the 7 patients brain volumes. Maximum Likelihood (ML $\equiv$ degenerate MRF), Logistic Regression (LR $\equiv$ degenerate DRF), SVM (degenerate SVRF), MRF, DRF, SVRF and DCRF. For each of the Random Field methods, we initialized inference with the corresponding degenerate classifier (*i.e.*, Maximum Likelihood, Logistic Regression, or SVM). To provide a fair comparison between SVM-based models (SVRF and DCRF) and the other models, we only used the linear kernel.

Table 2. Jaccard Percentage Scores for Enhancing Tumor and Edema Tumor Areas

Studies	Enhancing Tumor Area							Edema Area						
	ML	MRF	LR	DRF	SVM	SVRF	DCRF	ML	MRF	LR	DRF	SVM	SVRF	DCRF
1-1	23.1	24.6	44.4	46.1	50.7	52.8	**53.2**	21.9	21.6	35.7	36.7	58.0	**58.2**	58.0
2-1	0.0	0.0	61.3	61.5	87.4	**87.7**	87.1	33.3	34.2	59.2	61.4	**89.4**	89.2	89.3
3-1	69.2	69.7	61.8	61.8	83.0	84.8	**86.8**	34.4	34.4	75.5	77.2	81.7	**82.2**	81.9
3-2	40.1	40.3	84.8	84.6	85.7	**85.8**	**85.8**	47.6	48.1	73.6	74.1	80.3	**81.1**	80.5
4-1	26.9	27.3	49.1	50.4	78.8	81.7	**82.6**	28.3	29.1	38.6	41.2	54.0	**55.4**	54.6
4-2	58.9	59.7	68.3	70.2	76.7	77.9	**79.2**	43.2	46.8	45.3	46.7	54.7	**57.7**	54.9
4-3	49.2	50.2	71.3	71.6	88.2	88.1	**88.8**	35.4	35.4	69.9	**70.6**	69.2	69.1	69.1
4-4	65.6	68.2	**87.5**	87.1	87.0	87.1	86.9	44.1	43.7	78.6	79.0	77.7	77.3	**79.5**
5-1	67.0	67.5	52.2	51.4	82.8	**84.3**	84.1	47.8	48.6	63.6	65.7	74.8	**76.9**	74.6
6-1	37.4	37.6	76.4	76.2	79.2	**80.4**	80.0	40.3	40.1	79.3	79.7	82.2	**83.7**	82.9
7-1	63.2	63.0	75.5	76.7	81.0	**81.4**	81.1	74.9	77.7	91.2	92.4	94.8	**94.9**	**94.9**
7-2	37.7	39.3	75.9	75.8	86.5	**87.3**	86.8	39.2	40.4	80.9	82.7	**83.1**	82.8	**83.1**
7-3	45.3	45.6	81.8	81.5	87.7	87.6	**87.8**	54.1	53.9	79.3	80.7	84.6	84.5	**85.6**
Average	44.9	45.6	63.6	68.8	81.1	82.1	**82.3**	41.9	42.6	62.2	68.3	75.7	**76.4**	76.1

Table 3. Jaccard scores for Gross Tumor Areas

Studies	Gross Tumor Area						
	ML	MRF	LR	DRF	SVM	SVRF	DCRF
1-1	19.3	19.5	39.4	**40.9**	40.7	40.5	41.1
2-1	35.4	35.7	65.1	66.1	**78.2**	76.9	78.0
3-1	44.4	46.1	72.9	73.4	77.9	78.7	**78.2**
3-2	51.2	51.3	76.3	76.2	78.1	78.8	**80.2**
4-1	37.4	38.7	39.4	40.1	41.4	41.2	**42.1**
4-2	38.0	40.2	39.7	39.4	62.1	**64.9**	62.1
4-3	66.0	68.5	73.3	**73.5**	64.4	64.5	64.1
4-4	46.7	45.8	83.8	83.5	86.0	**87.0**	86.2
5-1	50.1	50.9	65.3	68.3	82.8	**84.8**	83.4
6-1	46.6	47.6	79.6	79.4	87.6	**88.2**	87.8
7-1	66.4	66.3	71.9	73.2	74.6	74.1	**74.7**
7-2	49.6	52.4	68.3	67.9	72.7	**72.9**	72.5
7-3	43.4	43.7	73.5	72.7	81.6	81.2	**82.0**
Average	45.7	46.7	60.6	65.7	71.4	**71.8**	71.7

The first task was the relatively easy one of segmenting the "enhancing" tumor areas — the region that appears hyper-intense after injecting a contrast agent. The second task was segmenting the entire edema area associated with the tumor; this is significantly more challenging due to the high degree of similarity between the intensities of edema areas and normal cerebrospinal fluid in the various modalities. The final task was segmenting the gross tumor area as defined by the radiologist. This can be a subset of the edema but a superset of the enhancing area, and is inherently a very challenging task even for human experts, given the modalities examined.

Tables 2 and 3 present the classification results for the three tasks. Over all three tasks, we see that the best results were typically obtained by either DCRF and SVRF, which were comparable to each other, and statistically better than the rest: The differences between SVRF and the next best, SVM, across the three tasks was significant at the $p < 0.000002$ level based on a paired example t-test, but the same t-test between SVRF and DCRF across the tasks indicates no difference — *i.e.*, here $p = 0.37$. However, Table 1 (second row) shows that our method requires significantly less training time — by a factor of 14! ($p <$ 2.285E-34) Although SVM performed very well visually on the three tasks, just as we saw on the synthetic data results, this performance can not always be guaranteed. In Table 2, the results from the second patient "2-1" produced an interesting observation; significant overlap between Gaussians in the high dimensional feature space leads ML and subsequently MRF to misclassify the entire area as non-tumor. This example shows that inappropriate modeling of $P(\mathbf{x} \mid \mathbf{y})$ can generate extremely poor performance. Although the segmentation tasks for edema and gross tumor areas are very hard, the best discriminative approaches (*i.e.*, SVRF and DCRF) still produce segmentations that are typically very similar to the manual segmentations, for all 3 tasks.

5 Conclusions

Learning to classify regions in an image is a challenging task, partly because labeling each pixel in an image can require modeling spatial correlations among neighboring pixels, which can be difficult to learn. As standard independent and identically distributed classification algorithms do not model these correlations, they typically fail to correctly classify data instances. Such spatial correlations can, however, be effectively modeled by various Random Field frameworks. However, these systems (especially the ones that work effectively.) can require a significant amount of time to learn. This time constraint makes such models inappropriate for large scale real-world problems, such as segmenting brain tumors.

In this paper, we have proposed a Decoupled CRF (DCRF) to improve the efficiency of a discriminative Random Field method for finding regions in an image. Our proposed model first learns the two potentials (Association and Local-consistency) *independently*, each based on a variant of Support Vector Machines. Afterwards, to segment regions in a novel image, it uses a new potential that is the simple sum of these potentials, using ICM (with respect to this combined potential) to produce a labeling. Our empirical results — on both synthetic and real-world data — show that this DCRF approach is virtually as accurate as the most accurate random field for this task (SVRF), but the learning time is many times faster (here, by a factor over 14 in one case, and over 30 in another). In addition, our model produces effective classification results, even when data sets are heavily imbalanced.

Acknowledgments. R. Greiner is supported by the National Science and Engineering Research Council of Canada (NSERC) and the Alberta Ingenuity Centre for Machine Learning (AICML). C.H. Lee is supported by AICML. O. Zaïane is supported by NSERC. Our thanks to Dale Schuurmans for helpful discussions on optimization and

parameter estimation, BTGP members for help in data processing, and Albert Murtha (M.D.) for domain knowledge on the tumor data set.

References

1. J. Besag. On the statistical analysis of dirty pictures. *Journal of Royal Statistical Society. Series B*, 48:3:259–302, 1986.
2. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *ICCV*, pages 377–384, 1999.
3. C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
4. T. G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
5. C. Garcia and J. Moreno. Kernel based method for segmentation and modeling of magnetic resonance images. *LNCS*, 3315:636–645, Oct 2004.
6. M. I. Jordan. *editor. Learning in Graphical Models*. MIT Press, 1999.
7. M. Kaus, S. Warfield, A. Nabavi, P. Black, F. Jolesz, and R. Kikinis. Automated segmentation of MR images of brain tumors. *Radiology*, 218:586–591, 2001.
8. R. Kindermann and J. Snell. Makrov random fields and their applications. *American Mathematical Society*, 1980.
9. S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. *NIPS*, 2003.
10. J. Lafferty, F. Pereira, and A. McCallum. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*, 2001.
11. C. Lee, M. Schmidt, and R. Greiner. Support vector random fields for spatial classification. *PKDD*, 2005.
12. S. Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, Tokyo, 2001.
13. H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on platt's probabilistic outputs for support vector machine. Technical report, 2003.
14. J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1999.
15. J. Platt. *Probabilistic outputs for support vector machines and comparison to regularized likelihood methods*. MIT Press, Cambridge, MA, 2000.
16. M. Schmidt. Automatic brain tumor segmentation. Master's thesis, University of Alberta, 2005.
17. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.
18. B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *ICML '04*, page 102, New York, NY, USA, 2004. ACM Press.
19. A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *NIPS 17*. MIT Press, Cambridge, MA, 2005.
20. J. Zhang, K. Ma, M. Er, and V. Chong. Tumor segmentation from magnetic resonance imaging by learning via one-class support vector machine. *International Workshop on Advanced Image Technology*, pages 207–211, 2004.

Group SAX: Extending the Notion of Contrast Sets to Time Series and Multimedia Data

Jessica Lin[1] and Eamonn Keogh[2]

[1] Information and Software Engineering
George Mason University
jessica@ise.gmu.edu
[2] Department of Computer Science & Engineering
University of California, Riverside
eamonn@cs.ucr.edu

Abstract. In this work, we take the traditional notation of *contrast sets* and extend them to other data types, in particular time series and by extension, images. In the traditional sense, contrast-set mining identifies attributes, values and instances that differ significantly across groups, and helps user understand the differences between groups of data. We reformulate the notion of contrast-sets for time series data, and define it to be the key pattern(s) that are maximally different from the other set of data. We propose a fast and exact algorithm to find the contrast sets, and demonstrate its utility in several diverse domains, ranging from industrial to anthropology. We show that our algorithm achieves 3 orders of magnitude speedup from the brute-force algorithm, while producing exact solutions.

1 Introduction

As noted by Bay and Pazzani, "*A fundamental task in data analysis is understanding the differences between several contrasting groups*". While there has been much work on this topic for discrete and numeric objects, to the best of our knowledge, the problem of mining contrast sets for time series or other multimedia objects has not been addressed before. This work makes two fundamental contributions to the problem.

- We introduce a formal notion of time series contrast set.
- We introduce a fast and exact algorithm to find time series contrast sets.

Contrast-set mining is a relatively new data-mining task, designed to identify differences among various groups of data. It can be roughly viewed as a variant of association rule mining [1, 4, 9]. While association rule mining discovers rules that describe or explain the current situation, contrast-set mining finds rules that differentiate contrasting groups of data, by identifying attributes and values (or conjunctions thereof) that differ meaningfully across them [2, 3]. Knowing these attributes that characterize the discrepancies across various groups can help users understand the fundamental differences among them, and make independent decisions on those groups accordingly.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 284 – 296, 2006.
© Springer-Verlag Berlin Heidelberg 2006

In this work, we introduce the concept of contrast-set mining for time series datasets. Due to the unique characteristics of time series data, the notion of contrast-set mining deviates from the traditional sense as defined on discrete data. More specifically, time series contrast-set mining aims to identify the key patterns rather than rules that differentiate two sets of data.

While this paper addresses *time series* contrast set explicitly, we note it is possible to convert many kinds of data into time series. For example, Figure 1 shows that we can convert shapes to time series. Other types of data, from text to video to XML [11], have also been converted into time series with varying degrees of success.

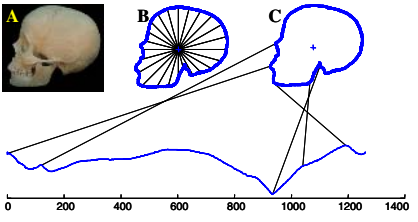


Fig. 1. Shapes can be converted to time series. A) A bitmap of a human skull. B) The distance from every point on the profile to the center is measured and treated as the Y-axis of a time series of length *n* (C) The mapping of the skull to time series.

Time series contrast-set mining should not be confused with clustering or summarization, in which the aim is to compare the *average* behavior of different sets. Suppose we are given two datasets that come from the same source; for example, two sets of time series telemetry from two shuttle launches, or two sets of ECG heartbeats from one patient – one before a drug was given and one after, or two sets of skulls, one set from Asia and one set from Europe. For some external reason, we suspect or actually know that there is something different about one group - maybe one shuttle crashed, or the patient had a heart attack. It might be the case that one entire group is different and this explains the problem. Maybe *all* the shuttle telemetry from the first launch has an amplitude scaling increase, or maybe *all* the ECG collected after the drug was administered show a faster heart rate. However, maybe a single or a few object(s) explain the difference. This is *exactly* what we are aiming to find.

As we shall show, our work has potential applications in detecting anomalies or differences in time series datasets in several diverse domains, and in certain images. The paper is organized as follows. In Section 2 we provide a brief overview on related work and necessary background. Section 3 describes the brute-force algorithm on finding time series contrast sets. In Section 4, we introduce an algorithm that offers 3 orders of magnitude speedup from the brute-force algorithm. Section 5 shows some experimental results on some time series datasets and image data. Section 6 concludes and offers suggestion for future work.

2 Related Work and Background

The most representative work on contrast-set mining is perhaps the work by Bay and Pazanni [2]. The authors introduce the task of contrast-set mining, and their algorithm STUCCO offers efficient search through the space of contrast-sets, based on the Max-Miner [4, 14] rule-discovery algorithm. Follow-up work by Webb et al. [14] discovered that existing commercial rule-finding system, Magnum Opus [13], can successfully perform the contrast-set task. The authors conclude that contrast-set mining is a special case of the more general rule-discovery task. He et al. [6] applies contrast-set mining on customer clusters to find cluster-defined actionable rules. In [10] Minaei-Bidgoli et al. proposed an algorithm to mine contrast rules on a web-based educational system. Their work is unique in that it allows rule discovery with very low minimum support, so rules that would have been otherwise overlooked can be discovered.

2.1 Notation

For simplicity and clarity we will speak only of time series below; however, as we hinted in Figure 1, and as we explicitly address in Section 5.2, these ideas can be extended to images and other types of data

Definition 1. *Time Series*: A time series $T = t_1,\ldots,t_m$ is an ordered set of m real-valued variables.

For data mining purposes, we are typically not interested in any of the *global* properties of a time series; rather, we are interested in *local* subsections of the time series, which are called subsequences.

Definition 2. *Subsequence*: Given a time series T of length m, a subsequence C of T is a sampling of length $n \leq m$ of contiguous position from p, that is, $C = t_p,\ldots,t_{p+n-1}$ for $1 \leq p \leq m - n + 1$.

Since all subsequences may potentially be attributing to contrast sets, any algorithm will eventually have to extract all of them; this can be achieved by use of a sliding window. We also need to define some distance measure $Dist(C,M)$ in order to determine the similarity between subsequences. We require that the function $Dist$ be symmetric, that is, $Dist(C,M) = Dist(M,C)$. There are dozens of distance measures for time series in the literature. However recent evidence strongly suggests that simple Euclidean distance is very difficult to beat for classification and clustering problems, so without loss of generality we will exclusively use it in this work.

Definition 3. *Euclidean Distance*: Given two time series Q and C of length n, the Euclidean distance between them is defined as:

$$Dist(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$$

Each time series subsequence is normalized to have mean zero and a standard deviation of one before calling the distance function, because it is well understood that in virtually all settings, it is meaningless to compare time series with different offsets and amplitudes [7].

We may wish to be able to find a pattern that occurs in one time series, but does not have a close match in another. Since such patterns could be used to differentiate time series, we call such patterns TS-Diffs.

Definition 4. *TS-Diff*: Given two time series T and S, and a user-given parameter n, *TS-Diff*(T,S,n) is the subsequence C of T, of length n, that has the largest distance to its closest match in S.

Note that this definition is not generally symmetric; that is, *TS-Diff*$(T,S,n) \neq$ *TS-Diff*(S,T,n). We may wish to be able to find the patterns that differentiate the two time series.

Definition 5. *Time Series Contrast Sets*: Given two time series T and S, a user given parameter n, let C = *TS-Diff(T, S, n)* and D = *TS-Diff(S, T, n)*. The contrast set CS for T and S is $\{C, D\}$.

Often there might be more than one pattern that differentiates two time series. The definition above can be easily extended to K time series contrast sets.

3 Finding Time Series Contrast Sets

We begin by describing the simple and obvious brute-force algorithm for finding the contrast sets between two time series T and S. For simplicity, let's assume that the two sets of data are of the same length, m, although in reality, their lengths need not be the same. Furthermore, let's assume that we are interested in finding patterns or subsequences of length n that differentiate the data. Since the results are not necessarily symmetric, we need to process T and S separately.

Intuitively, the definition of contrast sets tells us that the brute-force algorithm will need to compute the pairwise distances between all subsequences in T and S. That is, for each subsequence C in T, we need to compute its distance to all subsequences in S, in order to determine the distance to its closest match in S. The subsequence in T that has the greatest such value is then *TS-Diff* (T, S, n). This can be achieved with nested loops, where the outer loop considers each candidate subsequence in T, and the inner loop is a linear scan to identify the candidate's nearest match in S. The brute-force algorithm is easy to implement and produces exact results. However, its $O(m^2)$ time complexity makes this approach untenable for even moderately large datasets.

Fortunately, the following observations offer hope for improving the algorithm's running time. Recall what we wish to find here: we wish to identify the subsequence in T that is farther away from its nearest match in S than all other subsequences are from their respective nearest matches. Hence, we can keep track of the candidate that has the largest nearest match distance so far. This implies that we might not need to know the actual nearest match distance for every subsequence in T. The only piece of information that is crucial is whether or not a given subsequence in T can be the candidate for *TS-Diff*. This can be achieved by determining if the given subsequence has the *potential* of having a large nearest match distance.

Suppose we use the variable best_so_far_dist to keep track of the largest nearest-match distance so far. Consider the following scenario. Suppose after examining the first subsequence in T, we find that this subsequence is 10 units away from its nearest match in S (and we initialize best_so_far_dist to be 10). Now suppose as we move on to the next candidate subsequence in T, we find that it's 2 units away from the first

subsequence in *S*. At this point, we know that the current candidate could not be the candidate for *TS-Diff*, since its nearest match distance is *at most* 2 units, which is far less than best_so_far_dist. We can therefore safely abandon the rest of the search for this candidate's nearest match. In other words, when we consider a candidate subsequence *C* in *T*, we don't actually need to find its true nearest match in *S*. As soon as we find any subsequence in *S* that has a smaller distance to *C* than best_so_far_dist, we can abandon the search process for *C*, safe in the knowledge that it could not be *TS-Diff*.

Clearly, the utility of such optimization depends on the order in which the subsequences in *T* are considered, as well as the order in which the subsequences in *S* are matched against the current candidate. The earlier we examine a subsequence in *S* that has a smaller Dist(C, D) than best_so_far_dist, the earlier we can abandon the search process for the current candidate. This observation brings us to reducing the *TS-Diff* problem into a generic framework where, instead of examining all subsequences in sequential order, it allows one to specify any customized ordering of the subsequences to be examined. Table 1 shows the pseudocode.

Table 1. Heuristic TS-Diff Discovery

```
1   Function [dist, l]= Heuristic_Search(T, S, n, Outer, Inner)
2    best_so_far_dist = 0
3    best_so_far_TS = NaN
4
5   // Begin Outer Loop
6   For Each C in T ordered by heuristic Outer
7      nearest_neighbor_dist = infinity
8
9      // Begin Inner Loop
10     For Each D in S ordered by heuristic Inner
11        IF Dist(C, D) < best_so_far_dist
12           Break                    // Break out of Inner Loop
13        End
14        IF Dist(C, D) < nearest_neighbor_dist
15           nearest_neighbor_dist = Dist(C, D)
16        End
17     End                           // End Inner Loop
18     IF nearest_neighbor_dist > best_so_far_dist
19        best_so_far_dist = nearest_neighbor_dist
20        best_so_far_TS  = C
21     End
22  End                              // End Outer Loop
23  Return[ best_so_far_dist, best_so_far_TS]
```

We can consider the following to be the best scenario: for the ordering *Outer*, the subsequences in *T* are sorted by descending order of distances to their closest matches in *S*, so that the true *TS-Diff* is the first object examined, and best_so_far_dist is at its maximum value after the first iteration of the outer loop. For the ordering *Inner*, the subsequences in *S* are sorted in ascending order of distances to the current candidate *C* so that it's guaranteed that the first object examined in the inner loop will have a distance smaller than best_so_far_dist (otherwise *C* would have been placed towards the front of the queue). For this heuristic, the first invocation of the inner loop will run to completion to determine best_so_far_dist. Thereafter, all subsequent invocations of the inner loop will be abandoned after only one iteration, i.e. after discovering that the current distance is

smaller than best_so_far_dist. The total time complexity is thus O(m). This is the best possible scenario. We call this heuristic *magic*.

On the other hand, we should expect the worst-case scenario to be the exact opposite of the best-case scenario. In this scenario, the true *TS-Diff* will be the last object to be examined. More specifically, this heuristics has the worst-possible ordering such that for *Outer*, the subsequences in *T* are sorted by ascending order of distance to the closest match in *S*. For *Inner*, the time series in *S* are sorted in descending order of distance to the current candidate. In this case, we are back to the quadratic time complexity as the brute-force algorithm. This is the *perverse* heuristic.

Another possible strategy is to order the subsequences randomly for both *Outer* and *Inner* heuristics. Empirically it works reasonably well, and the inner loop is usually abandoned early, considerably speeding up the algorithm.

The three strategies discussed so far suggest that a linear-time algorithm is possible, but only with the aid of some very wishful thinking. The best-case heuristic requires perfect orderings of subsequences in the inner and outer loops. The only known way to produce such ordering is to actually compute the distances, which indirectly solve the problem. Even if the distances are known in advance, any sorting algorithm requires at least O(*mlogm*) time complexity. To ensure that the total running time is no worse than the worst-case running time, we must require a linear-time heuristic for *T* (outer loop, invoked once for the whole program), and a constant-time heuristic for every invocation of S ordering.

Observe that, however, for *Outer*, we do not actually need to achieve a perfect ordering to achieve dramatic speedup. All we really require is that among the first few candidate subsequences being examined, we have at least one that has a large distance to its closest match. This will give the best_so_far_dist variable a large value early on, which will allow more early terminations of the inner loop.

Similar observation goes for *Inner*. In the inner loop, we also do not actually need a perfect ordering to achieve dramatic speedup. We just need that among the first few subsequences in *S* being examined, we have at least one that has a distance to the current candidate that is smaller than the current value of best_so_far_dist. This is a sufficient condition to allow early termination of the inner loop.

We can imagine a full spectrum of algorithms, which only differ by how well they order subsequences relative to the best-case ordering. The random heuristic is somewhere between the best and the worst heuristics. Our goal then is to find the best possible approximations to the best-case heuristic ordering, which is the topic of the next section.

4 Group SAX: Approximating the Best-Case Heuristic

Our techniques for approximating the perfect ordering returned by the hypothetical best-case heuristic require us to discretize the real-valued time series data first. We choose **S**ymbolic **A**ggregate Appro**X**imation (SAX) representation of time series introduced in [8] to be the discretization technique. Since our algorithm finds differences between groups of time series using SAX, we call it *Group SAX*.

SAX works by first approximating the original time series of length *m* with *w* coefficients ($w << n$) via Piecewise Aggregate Approximation (PAA) [8]. These *w* coefficients are then converted to symbols of cardinality α, according to where they

reside in the Gaussian space. Therefore, the discrete approximation of the time series is a string of length *w*. Due to the space constraints, we direct interested readers to [8] for more information on SAX.

4.1 The *Outer* Loop Heuristic

We begin by creating two data structures to support our heuristics. Before that, there are two parameters associated with SAX that we must consider. They are the cardinality of the SAX alphabet size α, and the SAX word size *w*. Extensive experiments carried out by the current authors and dozens of other researchers worldwide [8] suggest that a value of either 3 or 4 for α is best for virtually any task on any dataset. Furthermore, while the choice of *w* depends on the data, we observe empirically that the speedup does not critically depend on *w*. We refer interested readers to [8] for more details on SAX parameter setting, but note that the parameters only affect the efficiency of the algorithm, not the final results. The subsequences are extracted by sliding a window of length n across the time series, which are then converted to SAX words. These SAX words are inserted to an array where the index refers back to the original sequence. Figure 2 gives a visual intuition of this, where both α and *w* are set to 3. Once we have this ordered list of SAX words, we construct a hash table. Each bucket in the hash table represents a unique word and contains a linked-list index of all subsequences that map to the corresponding string. The hash function we used assigns each SAX word a unique address ranging from 0 to $\alpha^w - 1$, hence guarantees minimal perfect hashing. The memory consumption for creating an empty hash table is considerably small and can be ignored.

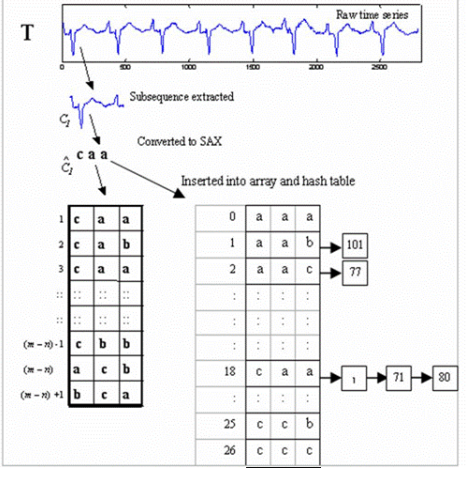


Fig. 2. The data structure used to support the *Inner* and *Outer* heuristics. (*left*) An array of SAX words (*right*). An excerpt of the hash table that contain pointers to the corresponding subsequences.

The process is repeated for both time series datasets that we wish to contrast. Each dataset has its own array and hash table. Once the hash tables are constructed for *T* and *S*, we can now determine the ordering of the outer loop objects. Recall that for *Outer*, the goal is to have a large nearest-match distance early in the loop. Thus, intuitively, we want to identify the subsequences in *T* that have the smallest count of matching SAX word in *S* (virtually zero). The reason is simple. Since SAX approximates and captures similarities between time series, we can expect that similar time series are likely to map to the same SAX representation. Conversely, subsequences mapped to SAX strings that are exclusive to *T*, i.e., they appear in *T* but not in *S,* are unlikely to have close matches in *S*. Therefore, by considering the candidate subsequences that map to such unique or rare SAX words with respect to S early in the outer loop, we have a great chance of obtaining large best_so_far_dist value among the first few candidates examined, thus allowing early termination in the subsequent inner-loop iterations.

To achieve this, we utilize the two hash tables we build from *T* and *S*. We perform a linear scan in *Hash_S*, and for each empty bucket we encounter, check if the corresponding bucket in *Hash_T* is empty as well. If not, then we record the subsequence indices from this bucket in *Hash_T*, and add them to a list which we call *Preferred_List*. We end up with a list that contains indices of subsequences whose SAX representations are unique in *T*. The subsequences referred to by the *Preferred_List* will be given to the outer loop to search over first. After the outer loop has exhausted this set, the rest of the candidates are visited in random order.

4.2 The *Inner* Loop Heuristic

Our *Inner* heuristic also leverages off *Hash_T* and *Hash_S*. Recall that in the inner loop, as soon as we encounter a subsequence similar enough to the current candidate in the outer loop, such that their distance is smaller than best_so_far_dist, then the search for the current candidate can be abandoned. The heuristic used for the outer loop gives us hope that best_so_far_dist will take on a large value early on in the process. For the inner loop, we take the optimization one step further by putting the subsequences that might cause early termination in the front of the queue so they are examined first. By identifying and eliminating those that could not possibly have a nearest match distance larger than best_so_far_dist early in the iteration, many unnecessary computations can be spared. Note that all it takes is having one distance that is smaller than best_so_far_dist.

To achieve this, we determine the ordering of the inner loop as follows. When candidate *i* is first considered in the outer loop, we look up the SAX word that it maps to, by examining the i^{th} word in the array for *T*. We then compute the key for the SAX word, and order the first items in the inner loop in the order of the elements in the linked list index found at the corresponding bucket in *Hash_S*. These subsequences will be visited first by the inner loop. After this list is exhausted, the rest of the subsequences are visited in random order.

Note that in the beginning, since the outer loop considers the unique words in *T* first, there will be no matching words in S, thus no optimization for the inner loop. One option would be to limit the size of *Preferred_List* in the outer loop; however, empirically we find that even without doing so, the speed up is already significant that it is not necessary to put a threshold on the number of “unique” subsequences to examine first in the outer loop.

5 Empirical Evaluation

5.1 The Utility of Contrast Sets: Time Series

We begin by considering the contrast set between German and Italian consumer electrical power demands. We obtained the last 9 nine years data from two towns, Dortmumd and Florence. Both time series are sampled hourly, and thus each is of length 269,379. We are interested in monthly patterns, so we set parameter *n* to 672 (4 weeks). Figure 3 shows the number one TS-Diff(*Italian*, *German*, 672) found.

The difference is striking. The German time series simply shows the typical weekly pattern repeated 4 times. The typical weekly pattern for power demand corresponds to 5 major peaks for the Monday to Friday 9 to 5 work hours, followed by a much smaller peak for Saturday and minimal power demand for Sunday. In contrast, the Italian demand shows a dramatic reduction in power demand as the month of August progresses. What is behind this difference?

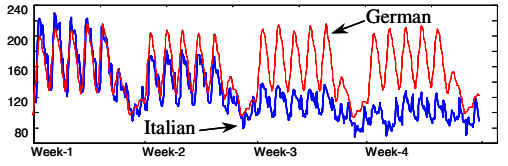


Fig. 3. The four weeks of Italian Power Demand, beginning on Monday July 31st 1995 is radically different from the *most* similar four weeks of German Power Demand, beginning on 3rd of May 1999.

The answer lies in an Italian cultural phenomenon. According to travel writer Nella Nencini, "By the middle of July, normal activity begins to wane and by the beginning of August, shops no longer close between 1 and 4 p.m., they close for two or three weeks. Dry cleaners close, mechanics close, factories close, wineries close, restaurants close, even some museums close. Cities like Florence and Venice would be abandoned if not for the tourists braving the heat to visit artistic treasures". The dramatic change in power demand reflects the fact that most major employers (like Fiat and many government offices) simple shut down for the month. This difference is obvious if we zoom out and look at a full year of Florence's power demand, as shown in Figure 4.

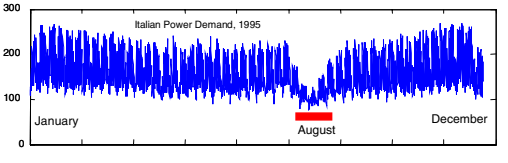


Fig. 4. One Year of Italian Power Demand (1995). Note that August is radically different from the rest of the year.

5.2 The Utility of Contrast Sets: Shapes

As noted in Section 1, we can trivially apply our ideas to shapes, since shapes typically represented as a 1-dimensional signal. In this section we show some examples of mining image datasets.

Petroglyphs are images incised in rock by prehistoric peoples. They were an important form of pre-writing symbols, used in communication from approximately 10,000 B.C. to modern times, depending on culture and location. Petroglyphs have been found on every continent except Antarctica. In virtually all cases, there is still great controversy about who created the petroglyphs, when, and for what purpose. The controversy is not for the want of evidence, for example the petroglyph shown in Figure 6A is just one of more than 100,000 petroglyphs to be found in an area of a mere 91 square miles of Sierra Nevada called Renegade Canyon.

We believe that contrast sets offer one possible tool for examining massive archives of petroglyph images. As a preliminary experiment we began by contrasting petroglyphs from the aforementioned site in Nevada with a similarly dense site in Sheep Springs in Kern County California.

For simplicity we only consider petroglyphs of animals, however it has be estimated that 51% of the Renegade Canyon petroglyphs are of animals, and as the name "Sheep Springs" suggests, the Californian site is similarly dense with images of sheep.

There are several technical problems that must be faced before we apply our algorithm. First of all there is the issue of image processing and shape extraction. This task is non-trivial, but not of direct interest here. Note that the representation we use is scale and translation invariant.

Once the shapes have been extracted and converted to time series we must consider two important issues. Do we wish to be enantiomorphic invariant? That is to say do we wish to attach any significance to whether an animal is facing left or right? After consulting with an anthropologist we decided to ignore such directional information. This we achieve by simply augmenting the database of time series to include the mirror image of each image. We can achieve this directly in the time series representation. Recall that a single time series C is defined as: $C = c_1,c_2,\ldots,c_j,\ldots,c_n$.

For each such time series we also add C' to the database: $C' = c_n,c_{n-1},\ldots,c_2,\ldots,c_1$.

We must also consider the problem of rotation invariance. Should we attach any significance to the angle at which the animals are drawn? For two reasons our answer is no. The first is pragmatic. Finding "correct" orientation of a shape is difficult problem in general. The second reason why we choose to be invariant to orientation is an observation by our anthropologist that often the animals are drawn to align with cracks and fissures in the rocks, and the orientation appears to have no significance.

Once again, achieving rotation invariance in our representation is easy to achieve by augmenting our database to contain additional time series. Let $\mathbf{C}$ be all n circular shifts of C:

$$\mathbf{C} = \begin{Bmatrix} c_1, c_2, \ldots, c_{n-1}, c_n \\ c_2, \ldots, c_{n-1}, c_n, c_1 \\ \vdots \\ c_n, c_1, c_2, \ldots, c_{n-1} \end{Bmatrix}$$

By adding all such circular shifts to our database we achieve rotation invariance.

Figure 5 shows some examples of shapes from Renegade Canyon and their closest match from Sheep Springs, including the one that maximizes TS_Diff(Renegade Canyon, Sheep Springs).

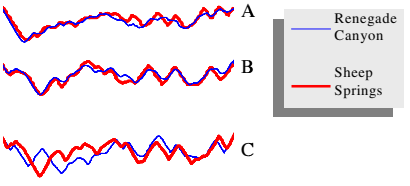


Fig. 5. Three petroglyphs shapes from Renegade Canyon (light lines) and their best matches from the Sheep Springs database. While most shapes are like A and B in having a close match in the other database, the shape from Renegade Canyon shown in C is unusually different from its nearest counterpart in Sheep Springs.

Why is one image from Renegade Canyon so different to any image from Sheep Springs? An inspection of the original images, as shown in Figure 6, reveals the answer. While there are a handful of images that show a spear stuck into the body of a sheep in Renegade Canyon, including the one shown in Figure 6 (top), a careful manual inspection of the Sheep Springs database reveals that there are *no* such petroglyphs in Sheep Springs.

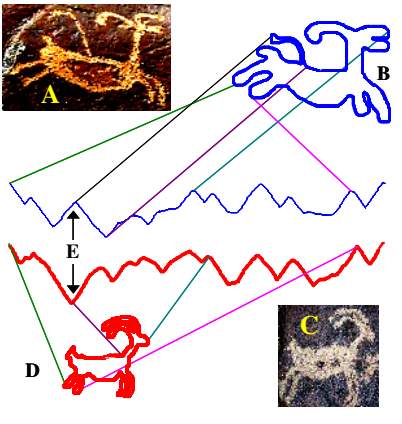


Fig. 6. A) A petroglyph from Renegade Canyon of a Bighorn Sheep with a spear stuck in it has its profile automatically extracted (B) and converted to a time series. C) A petroglyph from Sheep Springs of a Bighorn Sheep with a spear stuck in it has it profile automatically extracted (D) and converted to a time series. Note that the two shapes are not a good match, mainly because the part of the time series corresponding to the spear (E) creates a relatively large Euclidean distance between the two shapes.

5.3 The Utility of Our Search Technique

In Figure 7, we compare the brute force algorithm to our algorithm in terms of the number of times the Euclidean distance function is called. Since we are now

interested in the scalability of both approaches, we use random walk datasets of lengths 100,000 and 200,000. For our algorithm we averaged the results over 100 runs for each data length.

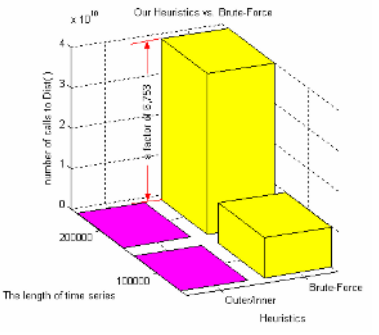


Fig. 7. The number of calls to the distance function required by brute force and heuristic search. The window length is 64 for all cases.

Note that as the data sizes increase, the differences get larger. For a time series of length 200,000, our approach is almost seven thousand times faster than brute force approach. This experiment is in fact pessimistic in that we used the datasets (random walk) that did not have any obvious structures in them. In general, if the data exhibit some structures, as the ones used in Section 5.1 and 5.2, then our approach would be even faster since there would be a lot more potential matches for most subsequences, to allow early termination.

6 Conclusions and Future Work

In this work, we have introduced the notion of time series contrast sets, a data mining task that identifies key differences across data groups. We introduced an algorithm to efficiently find time series contrast sets and demonstrated its utility of a host of domains. Many future directions suggest themselves; most obvious among them are extensions to multidimensional time series, to streaming data, and to other distance measures. We will also investigate the possibility of combining the processes of *TS-Diff(T, S, n)* and *TS-Diff(S, T, n)* to avoid redundant computations.

References

[1] Agrawal, R., Imielinski, T. & Swami, A. (1993). Mining Associations Between Sets of Items in Massive Databases. In proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 207-216.

[2] Bay, S. (2000). Multivariate Discretization of Continuous Variables for Set Mining. In proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Boston, MA. Aug 20-23.

[3] Bay, S. D. & Pazzani, M. J. Detecting Change in Categorical Data: Mining Contrast Sets (1999). In proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 15-18. San Diego, CA. 302-306.

[4] Bayardo, J. & Roberto, J. (1998). Efficiently Mining Long Patterns from Databases. In proceedings of 1998 ACM SIGMOD International Conference on Management of Data. pp. 85-93.

[5] Bentley. J. L. & Sedgewick. R. (1997). Fast algorithms for sorting and searching strings. In Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 360-369

[6] He, Z., Xu, X. & Deng, S. (2004). Mining Cluster-Defining Actionable Rules. In proceedings of NDBC '04.

[7] Keogh, E. & Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In Proc. of SIGKDD. pp 102-111.

[8] Lin, J., Keogh, E., Lonardi, S., & Chiu, B. (2003). A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.

[9] Menzies, T. & Hu, Y (2003). Data Mining for Very Busy People. IEEE Computer, October, pp. 18-25.

[10] Minaei-Bidgoli, B., Tan, P-N., & Punch, W. F. (2004). Mining Interesting Contrast Rules for a Web-based Educational System. In proceedings of 2004 International Conference on Machine Learning Application. Louisville, KY. Dec 16-18.

[11] Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, Andrea Pugliese (2005). Fast Detection of XML Structural Similarity. IEEE Trans. Knowl. Data Eng. 17(2): 160-175.

[12] Tanaka, Y. & Uehara, K. (2004). Motif Discovery Algorithm from Motion Data. In proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI).

[13] Webb, G. (2001). Magnum Opus version 1.3. Computer software, Distributed by Rulequest Research.

[14] Webb, G., Butler, S. & Newlands, D. (2003). On Detecting Differences Between Groups. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington DC. Aug 24-27.

An Attacker's View of Distance Preserving Maps for Privacy Preserving Data Mining

Kun Liu, Chris Giannella, and Hillol Kargupta*

Department of Computer Science and Electrical Engineering,
University of Maryland Baltimore County,
1000 Hilltop Circle, Baltimore, MD 21250, USA
{kunliu1, cgiannel, hillol}@cs.umbc.edu

Abstract. We examine the effectiveness of distance preserving transformations in privacy preserving data mining. These techniques are potentially very useful in that some important data mining algorithms can be *efficiently* applied to the transformed data and produce *exactly the same* results as if applied to the original data *e.g.* distance-based clustering, k-nearest neighbor classification. However, the issue of how well the original data is hidden has, to our knowledge, not been carefully studied. We take a step in this direction by assuming the role of an attacker armed with two types of prior information regarding the original data. We examine how well the attacker can recover the original data from the transformed data and prior information. Our results offer insight into the vulnerabilities of distance preserving transformations.

1 Introduction

Recent interest in the collection and monitoring of data using data mining technology for the purpose of security and business-related applications has raised serious concerns about privacy issues. For example, mining health-care data for detection of bio-terrorism may require analyzing clinical records and pharmacy transaction data of certain off-the-shelf drugs. However, combining such diverse data sets belonging to different parties may violate privacy laws. Privacy Preserving Data Mining (PPDM) strives to provide a solution to this dilemma. It aims to allow useful data patterns to be extracted without compromising privacy.

Data perturbation represents one common approach in PPDM. Here, the original dataset is perturbed and the result is released for data analysis. Perturbation approaches typically face a "privacy/accuracy" trade-off. On the one hand, perturbation must not allow the original data records to be adequately recovered. On the other, it must allow "patterns" in the original data to be recovered. In many cases, increased privacy comes at the cost of reduced accuracy and vice versa. For example, Agrawal and Srikant [1] proposed adding randomly generated i.i.d. noise to the dataset. They showed how the distribution from which the original data arose can be estimated using only the perturbed data. However, Kargupta *et al.* [2] and Huang *et al.* [3] pointed out how, in many cases,

* Also affiliated with AGNIK, LLC, USA.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 297–308, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the noise can be filtered off leaving a reasonably good estimation of the original data. These results point to the fact that unless the variance of the additive noise is sufficiently large, original data records can be recovered unacceptably well. However, this increase in variance reduces the accuracy with which the original data distribution can be estimated. This privacy/accuracy trade-off is not limited to additive noise, some other perturbation techniques suffer from a similar problem *e.g.* k-anonymity [4].

Recently, distance preserving data perturbation [5,6] has gained attention since it mitigates the privacy/accuracy trade-off by guaranteeing perfect accuracy. Many important data mining algorithms can be *efficiently* applied to the transformed data and produce *exactly the same* results as if applied to the original data. *e.g.* distance-based clustering and k-nearest neighbor classification. However, the issue of how well the original data is hidden has, to our knowledge, not been carefully studied. In this paper, we address this issue by studying how well an attacker can recover the original data from the transformed data and prior information. We restrict our attention to the class of distance preserving transformations that fix the origin and consider recovery of the original data in the presence of two different classes of prior information (described later). Our analysis explicitly illuminates scenarios where privacy can be breached. As such, valuable information is gained into the effectiveness of distance preserving transformation for privacy preserving data mining.

The remainder of this paper is organized as follows. Section 2 discusses some basic mathematical properties of distance preserving transformations, the application of these transformations to privacy-preserving data mining, and two classes of attacker prior knowledge. Sections 3 and 4 examine in detail how knowledge in each of these classes can be used to estimate the original data from the transformed data. Section 5 discusses related work. Finally, section 6 concludes the paper with a brief discussion of a suggested remedy for the attacker's approach in one of the classes of prior knowledge.

2 Distance Preserving Transformations

Throughout this paper (unless otherwise stated), all matrices and vectors discussed are assumed to have real entries. All vectors are assumed to be column vectors and M' denotes the transpose of any matrix M. An $m \times n$ matrix M is said to be orthogonal if $M'M = I_n$, the $n \times n$ identity matrix.[1] Let $\mathbb{O}_n$ denote the set of all $n \times n$, orthogonal matrices. A function $T : \mathbb{R}^n \to \mathbb{R}^n$ is distance preserving if for all $x, y \in \mathbb{R}^n$, $||x - y|| = ||T(x) - T(y)||$, where $||.||$ denotes l^2-norm of a vector. Here T is also called a *rigid motion*. It has been shown that any distance preserving transformation is equivalent to an orthogonal transformation followed by a translation [7, pg. 128]. In other words, there exists $M_T \in \mathbb{O}_n$ and $v_T \in \mathbb{R}^n$ such that T equals $x \in \mathbb{R}^n \mapsto M_T x + v_T$. If T fixes the origin, $T(0) = 0$, then $v_T = 0$, hence, T is an orthogonal transformation. Henceforth we assume T is a distance preserving transformation which fixes the origin – an

[1] If M is square, it is orthogonal if and only if $M = M^{-1}$ [7, pg. 17].

orthogonal transformation. Next we describe the privacy application scenarios where orthogonal transformation can be used to hide the data while allowing important patterns to be discovered *without error*.

2.1 Privacy Application Scenarios

We consider two privacy application scenarios as follows.

Census scenario: An organization has a private dataset X (each column is a data record) and wishes to make it publicly available for data analysis while keeping the original data records private. To accomplish this, $Y = M_T X$ is released to the public. The distance preserving nature of T allows a public entity to easily recovery many useful patterns from Y. For example, the cluster membership produced by a Euclidean distance-based K-means clustering on Y will be exactly the same as that produced on X. This model is widely studied in the field of security control for statistical databases. We refer the reader to [8] for a nice overview on this topic.

Storage outsourcing scenario: An organization continuously generates private data records, but does not wish to invest in the infrastructure (both personnel and hardware) needed to manage the storage. Outsourcing this job can be an attractive alternative *i.e.* the data records are handed over to an outside agency who manages their storage. However, the original data records are sensitive and the organization would rather avoid releasing them in the plain to the outsourcing agency. To accomplish this, the owner applies T to each data record and releases the results to the outsourcing agency. Whenever the owner wishes to retrieve records from the outsourced database, she transforms her query by the same T and sends it to the outsourcing agency who carries out similarity comparison on the data and, in turn, sends the results back to the owner. This scenario is closely related to work on secure database outsourcing, *e.g.* [17].

2.2 Prior Knowledge

Let the $n \times m$ matrix X denote a private dataset, with each column of X being a record and each row an attribute. We assume that the attacker knows that T is an orthogonal transformation and knows the perturbed data $Y = M_T X$. In most realistic scenarios, the attacker has some additional *prior knowledge* which can potentially be used effectively for breaching privacy. We consider two types of prior knowledge.

Known input-output: The attacker knows some collection of linearly independent private data records. In other words, the attacker has a set of linearly independent input-output pairs.

Known sample: The attacker knows that the original dataset arose as independent samples of some n-dimensional random vector V with unknown p.d.f. Also the attacker has another collection of independent samples from V. For technical reasons, we make a mild additional assumption: the covariance matrix of V has distinct eigenvalues.

In the next two sections, we describe and analyze an attack technique for *each type of* prior knowledge listed above.

3 Known Input-Output Attack

Let X_k denote the first k columns of X and X_{m-k} the remainder (likewise for Y). We assume that columns of X_k are all linearly independent and X_k is known to the attacker (Y is, of course, also known). The goal of the attacker is to recover some columns in X_{m-k} with at most $\epsilon \geq 0$ error (described later). If $k = n$, then the attacker can recover X_{m-k} perfectly as it equals $(Y_k X_k^{-1})' Y_{m-k}$. Thus, we assume $k < n$. Based on known information, the attacker can narrow down the space of possibilities for M_T to $\mathbb{M}(X_k, Y_k) = \{M \in \mathbb{O}_n : MX_k = Y_k\}$. Since the attacker has no additional information, any of these matrices is equally likely to have been M_T. The attacker chooses $\hat{M}$ uniformly from $\mathbb{M}(X_k, Y_k)$ and chooses index $1 \leq \hat{i} \leq m-k$ using some criterion (described later), then produces $\hat{x} = \hat{M}' y_{\hat{i}} = \hat{M}' M_T x_{\hat{i}}$ as an estimate of $x_{\hat{i}}$, where $x_{\hat{i}}$ is the $\hat{i}^{th}$ column of X_{m-k}. We say that an *ϵ-privacy breach* occurs if $||\hat{x} - x_{\hat{i}}|| \leq ||x_{\hat{i}}||\epsilon$. We define $\rho(x_{\hat{i}}, \epsilon)$ as the probability that an *ϵ-privacy breach* occurs. This serves as the criterion for choosing $\hat{i}$.

Next, for any vector $x \in \mathbb{R}^n$, we develop a closed form expression for $\rho(x, \epsilon)$, the probability that $||\hat{M}' M_T x - x|| \leq ||x||\epsilon$. This is the *$\epsilon$-privacy breach* probability for x. Due to space limitations, all proofs are omitted.

3.1 Probability of Privacy Breach

Let $Col(X_k)$ denote the column space of X_k and $Col_\perp(X_k)$ denote its orthogonal complement, *i.e.* $\{z \in \mathbb{R}^n : z'w = 0, \forall w \in Col(X_k)\}$. Since the columns of X_k are linearly independent, then there exists orthogonal matrices U_k $(n \times k)$ and U_{n-k} $(n \times (n-k))$ such that $Col(X_k) = Col(U_k)$ and $Col_\perp(X_k) = Col(U_{n-k})$. It can be proved that

$$\mathbb{M}(X_k, Y_k) = \{M_T U_k U_k' + M_T U_{n-k} P U_{n-k}' : P \in \mathbb{O}_{n-k}\}.$$

Hence, linear map $L : M \in \mathbb{M}(X_k, Y_k) \mapsto (M_T U_{n-k})' M U_{n-k} \in \mathbb{O}_{n-k}$ is a bijection. It can be further shown that

$$||\hat{M}' M_T x - x|| = ||L(\hat{M})' U_{n-k}' x - U_{n-k}' x||.$$

Thus, $\rho(x, \epsilon)$ equals the probability that a matrix $\hat{P}$ drawn uniformly from $\mathbb{O}_{n-k}$ satisfies

$$||\hat{P}' U_{n-k}' x - U_{n-k}' x|| \leq ||x||\epsilon. \tag{1}$$

Now let $S_{n-k}(U_{n-k}' x)$ be the hypersphere in $\mathbb{R}^{n-k}$ centered at the origin with radius $||U_{n-k}' x||$. Vector $\hat{P}' U_{n-k}' x$ and $U_{n-k}' x$ from inequality (1) are points on the surface of $S_{n-k}(U_{n-k}' x)$. Let $S_{n-k}(U_{n-k}' x, ||x||\epsilon)$ be the portion of S_{n-k} whose distance from $U_{n-k}' x$ is no larger than $||x||\epsilon$, *i.e.* $S_{n-k}(U_{n-k}' x, ||x\epsilon||) =$

$\{z \in S_{n-k}(U'_{n-k}x) : ||z - U'_{n-k}x|| \leq ||x||\epsilon\}$. From inequality (1), it follows that $\rho(x, \epsilon)$ is the probability that a randomly chosen $\hat{P} \in \mathbb{O}_{n-k}$ satisfies $\hat{P}'U'_{n-k}x \in S_{n-k}(U'_{n-k}x, ||x||\epsilon)$. Therefore, this probability equals the ratio of the surface area of $S_{n-k}(U'_{n-k}x, ||x||\epsilon)$ to the surface area of $S_{n-k}(U'_{n-k}x)$. Then, it can be shown:

$$\rho(x, \epsilon) = (\frac{1}{\pi})2arcsin(\frac{||x||\epsilon}{2||U'_{n-k}x||}) \text{ if } ||x||\epsilon < 2||U'_{n-k}x||; \text{ 1 otherwise.}$$

An alternate characterization of $||U'_{n-k}x||$ yields a more intuitive form of the second right-hand side. Consider $U_kU'_kx$ the projection of x into $Col(X_k)$. The distance, $d(x, X_k)$, of x from $Col(X_k)$ is $||x - U_kU'_kx||$. It can be shown that $||U'_{n-k}x|| = d(x, X_k)$. Therefore,

$$\rho(x, \epsilon) = (\frac{1}{\pi})2arcsin(\frac{||x||\epsilon}{2d(x, X_k)}) \text{ if } ||x||\epsilon < 2d(x, X_k); \text{ 1 otherwise.} \quad (2)$$

This formula allows us to observe the behavior of the ϵ-*privacy breach* probability for x in terms of $||x||\epsilon$ and the distance of x from $Col(X_k)$. Indeed the probability is approximately inversely proportional to $d(x, X_k)$ for $d(x, X_k) >> ||x||\epsilon$. [2] On the other hand, as $||x||\epsilon \to 2d(x, X_k)$, the breach probability goes to one. In the extreme case where $x \in Col(X_k)$, a breach occurs with probability 1 for any ϵ.

3.2 Attack Technique

Using equation (2), $\rho(x_{\hat{i}}, \epsilon)$ can be computed from $||x_{\hat{i}}||$, ϵ, and $d(x_{\hat{i}}, X_k)$. Since the attacker knows Y, she can compute $||y_{\hat{i}}|| = ||M_Tx_{\hat{i}}|| = ||x_{\hat{i}}||$ and V_k an $n \times k$, orthogonal matrix such that $Col(V_k) = Col(Y_k)$. It can be shown that $d(x_{\hat{i}}, X_k) = d(y_{\hat{i}}, Y_k) = ||M_Tx_{\hat{i}} - VV'M_Tx_{\hat{i}}||$. Therefore, the attacker chooses $\hat{i}$ to maximize $\rho(x_{\hat{i}}, \epsilon)$. If the data owner knows that X_k is in the attacker's prior knowledge, then the owner can protect against this attack by simply not releasing M_Tx_i for any x_i where $d(x_i, X_k)$ is unacceptably small. On the other hand, if the owner does not know X_k is prior knowledge, then this attack technique can be quite damaging.

4 Known Sample Attack

In this scenario, we assume that each data record arose as an independent sample from a random vector V with unknown p.d.f. We also make the following mild technical assumption: the population covariance matrix Σ_V of V has all distinct eigenvalues. We make this assumption because it holds in most practical situations [9, pg. 27]. Furthermore, we assume that the attacker has a collection of p samples that arose independently from V – these are denoted as the columns

[2] For small z, $arcsin(z)$ is approximately linear.

of matrix S. In this section we design a Principal Component Analysis (PCA)-based attack technique by which the attacker produces $\hat{X}$, an estimate of X, from $Y = M_T X$ and S. Unlike Section 3, we do not attempt a rigorous analysis of the attacker's success probability. Instead, we analyze the recovery error through experiments.

4.1 PCA Preliminaries

Let Σ_V denote the population covariance matrix of V. Since Σ_V is an $n \times n$, symmetric matrix (and we assume it has all distinct eigenvalues), it has n real eigenvalues $\lambda_1 > \ldots > \lambda_n$ [10, pg. 295]. Associated with each eigenvalue λ_i is its eigenspace, $\{z \in \mathbb{R}^n : \Sigma_V z = z\lambda_i\}$. It can be shown that since Σ_V has distinct eigenvalues, the eigenspaces are pair-wise orthogonal and each has dimension one [10, pg. 295]. As is standard practice, we restrict our attention to only a small number of eigenvectors. Let $\mathcal{Z}(V)_i$ denote the set of all eigenvectors $z \in \mathbb{R}^n$ such that $\Sigma_V z = z\lambda_i$ and $||z|| = 1$. Now consider random vector $T(V) = M_T V$ and let $\Sigma_{M_T V}$ denote its covariance matrix. The eigenspaces of Σ_V are related in a natural way to those of $\Sigma_{M_T V}$, as shown by the following theorem (all proofs are omitted due to space constraints).

Theorem 1. *The eigenvalues of Σ_V and $\Sigma_{M_T V}$ are the same and $M_T \mathcal{Z}(V)_i = \mathcal{Z}(M_T V)_i$ where $M_T \mathcal{Z}(V)_i$ equals $\{M_T w : w \in \mathcal{Z}(V)_i\}$.*

Since all the eigenspaces of Σ_V have dimension one, it can be shown that $\mathcal{Z}(V)_i$ contains only two eigenvectors $z_i, -z_i$, *i.e.* $\mathcal{Z}(V)_i = \{z_i, -z_i\}$. Let z_i be the lexicographically larger vector among $z_i, -z_i$, and let Z be the $n \times n$ matrix whose i^{th} column is z_i. Since the eigenspaces of Σ_V are pairwise orthogonal and $||z_i|| = 1$, Z is orthogonal. Similarly, we have that $\mathcal{Z}(M_T V)_i = \{w_i, -w_i\}$ (w_i is the lexicographically larger among $w_i, -w_i$) and W is the matrix with i^{th} column w_i (W is orthogonal). The following result forms the basis of the attacker's attack algorithm.

Corollary 1. *Let $\mathbb{I}_n$ be the space of all $n \times n$, matrices with each diagonal entry ± 1 and each off-diagonal entry 0 (2^n matrices in total). There exists $D_0 \in \mathbb{I}_n$ such that $M_T = W D_0 Z'$.*

4.2 PCA Attack Algorithm

First assume the attacker knows the population covariance Σ_V and $\Sigma_{M_T V}$. Thus, the attacker can compute W and Z'. By Corollary 1, the attacker knows that M_T equals $W D_0 Z'$ for some $D_0 \in \mathbb{I}_n$, and therefore, the original data would be recovered by $M_T' Y = Z D_0 W' Y$. The problem is how to choose the right D_0 from all the possible 2^n elements in $\mathbb{I}_n$. To do so, the attacker must utilize S and Y, in particular, the fact that these arose as independent samples from V and $M_T V$, respectively. For each $D \in \mathbb{I}_n$, each column of $WDZ'S$ arose as an independent sample from $WDZ'V$. If $D = D_0$, then $WDZ' = M_T$, so, $WDZ'S$ and Y should come from the same p.d.f. The attacker will choose $D \in \mathbb{I}_n$ such

that $WDZ'S$ is most likely to have arisen from the same p.d.f. as Y. To make this choice, a similarity function $G(WDZ'S, Y)$ is introduced, and the D that maximizes G is chosen. There might be many ways to define this function. In this paper, we use a multivariate two-sample hypothesis test for equal distributions [11]. The two-sample problem assumes that there are two sets of independent samples $x_1, x_2, \ldots, x_{m1}$ and $y_1, y_2, \ldots, y_{m2}$ of independent random vectors with distributions F_1 and F_2, respectively. The goal of two-sample problem is to test $H_0 : F_1 = F_2$, versus the composite alternative $H_1 : F_1 \neq F_2$. For each $D \in \mathbb{I}_n$, we compute the p-value of the test on $WDZ'S$ and Y, denoted by $\rho(D)$. Here the p-value is defined as the smallest level of significance at which H_0 would be rejected on a given data set. Small p-values suggest that the null hypothesis is unlikely to be true. The smaller it is, the more convincing is the rejection of the null hypothesis. Therefore the value of function G is nothing but the p-value, and the D matrix that is associated with the highest p-value is chosen.

In practice, the population covariance Σ_V and Σ_{M_TV} are unknown, and will be replaced by the sample covariance Σ_S and Σ_Y from S and Y (independent samples arising from V and M_TV). Algorithm 4.2.1 shows the complete PCA-based attack procedure.

Algorithm 4.2.1. PCA-based Attack Technique

Inputs: S, an $n \times p$ matrix where each column arose as an independent sample from V (a random vector with unknown p.d.f). $Y = M_TX$ where M_T is an unknown, $n \times n$, orthogonal matrix; and X is an $n \times m$ unknown matrix where each column arose as an independent sample from V.

Outputs: $\hat{X}$, an estimation of X.

Assumptions: Σ_V has all distinct eigenvalues.

1: Compute sample covariance matrix $\hat{\Sigma}_S$ from S and sample covariance matrix $\hat{\Sigma}_Y$ from Y. $[O(n^2m + n^2p)]$
2: Compute the eigenvector matrix $\hat{Z}$ of $\hat{\Sigma}_S$ and $\hat{W}$ of $\hat{\Sigma}_Y$. Each eigenvector has unit length and is sorted in the matrix by the corresponding eigenvalue. $[O(n^3)]$
3: Choose $D_0 = argmax\{G(\hat{W}D\hat{Z}\ S, Y) : D \in \mathbb{I}_n\}$. $[O(2^nB)]$
4: Compute $\hat{X} = \hat{Z}D_0\hat{W}\ Y$. $[O(n^3 + n^2m)]$

The computation cost of Algorithm 4.2.1 is $O(n^2(m+p)+n^3+2^nB)$ assuming $G(.,.)$ requires $O(B)$ computation. For the two-sample test, $B = (m+p)^2$, so, the total computation of the algorithm is $O(2^n(m+p)^2)$.

4.3 Effectiveness

The effectiveness of the PCA Attack algorithm depends on two correlated aspects: 1) the p.d.f., f, of V; and 2) the quality of covariance estimation.

PDF of V: First, suppose for some $D_1 \neq D_0 \in \mathbb{I}_n$, f is *invariant over* D_1 in the sense that $f_{D_1} = f_{D_0}$ where f_{D_i} is the p.d.f. $x \in \mathbb{R}^n \mapsto f(WD_iZ'x)$. Then,

$WD_0Z'S$, $WD_1Z'S$ and Y all arose from the same p.d.f., so $\rho(D_0)$ may not be larger than $\rho(D_1)$, and the attack algorithm will fail. An example of such an f is the n-variate Gaussian with mean vector zero and covariance matrix I_n. This distribution is invariant to orthogonal transformation. Second, suppose the eigenvalues of Σ_V are nearly identical. For example, suppose f has a diagonal covariance matrix whose diagonal entries (from top-left to bottom-right) are d, $d-\epsilon$, $d-2\epsilon$, ..., $d-n\epsilon$ where $d-n\epsilon > 0$ and $0 < \epsilon < 1$. Small errors in estimating Σ_V from S can produce a different ordering of the eigenvectors, hence, large errors in the attacker's recovery.

Quality of Covariance Estimation: A great deal of work has been conducted in the statistics community on estimating the covariance matrix of a random vector based on an independent sample [9, Chapter 10.4]. Any estimation technique can be used in our technique. In experiments we use the simple, standard sample covariance estimator.

4.4 Experiments

To validate the PCA-based attack algorithm, we conducted experiments on both synthetic and real world data. One such synthetic dataset contains 1000 data points, which are generated from a two-dimensional Gaussian distribution with mean $(-10, 10)$ and covariance $\begin{pmatrix} 1 & 1.5 \\ 1.5 & 3 \end{pmatrix}$. The attacker has 50 sample data points (5% of the size of original data) chosen from the same distribution. Figure 1 shows the results of perturbation and recovery. It can be seen that although the perturbed data is very different from the original one, the recovered data almost overlaps with the original data.[3] To further examine how sample size affects the quality of the attack, we fixed the orthogonal perturbation matrix, and variated the number of samples from 1% of the original data to 20%. For each sample ratio, 20 independent trials were conducted. We computed 95% confidence interval of the results. Figure 2 shows that as the sample size increases, the average relative distance between the columns of X and $\hat{X}$ decreases.[4]

For real world data, we chose the Adult Database from the UCI machine learning repository. This data set contains 32,561 records, and it is extracted from the census bureau database. For the purpose of visualization, we only selected three continuous attributes: age, education-num and hours-per-week, for the experiment. We first randomly separated the dataset into two disjoint sets. One set is viewed as the original data, and the other one is the attacker's sample data, which accounts for 5% of the original data. The left column of Figure 3 shows the difference between the original data and the perturbed data; the right column of Figure 3 depicts the results of PCA-based attack. It can be seen that the recovered data approximates the original data very well. To examine the influence of sample size, we fixed the orthogonal perturbation matrix, and

[3] Note that the shape of the perturbed data does not appear very similar to the shape of the original data because the axes scales are not even.

[4] The average relative distance between the columns is defined as $\frac{\sum_i^{numCols} \frac{||X_i - \hat{X}_i||}{||X_i||}}{numCols}$.

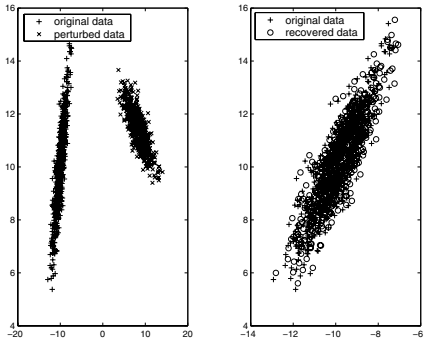


Fig. 1. Performance of PCA-based attack for two-dimensional Gaussian data

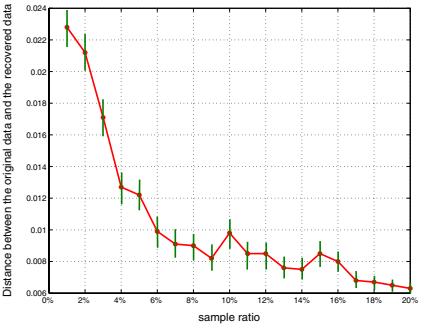


Fig. 2. Performance (average of 20 independent trials) w.r.t. sample size. Error bars show 95% confidence intervals.

variated the number of samples from 2% of the original data to 20%. For each sample ratio, 20 independent trials were conducted. Figure 4 gives the result.

To evaluate the complexity of the PCA attack algorithm, we generated multivariate Gaussian data with dimensionality ranging from 2 to 12. Each data set contains 5250 records, 250 records of which are used as samples. The energy test proposed in [11] was used to quantify similarity ($G(.,.)$), The experiment was conducted in Matlab on a dual-processor workstation with 3.00GHz and 2.99GHz Xeon CPUs and 3.00GB RAM. We observed that for 2-dimensional data, it took 143.1090 seconds, and for 12-dimensional data, it took 1.2442×10^5 seconds. Although the running time goes up rapidly as the dimension increases, this algorithm is still computationally feasible for relatively high dimensional data.

5 Related Work

This section presents a brief overview of the literature on data perturbation for PPDM. There is another class of PPDM technique using secure multi-party computation (SMC) protocols for implementing common data mining algorithms across distributed datasets. We refer interested readers to [12] for more details.

Additive perturbation: Agrawal and Srikant [1] proposed the addition of i.i.d., white noise for privacy protection. They describe a technique by which the original data distribution can be estimated from the perturbed data. Kargupta *et al.* [2] questioned the use of additive, white noise by showing how, in some cases, the noise can be effectively filtered off revealing a good approximation of the original data. This technique was further investigated by Huang *et al.* [3]. To our knowledge, these techniques are not applicable to this paper since it is concerned with non-additive perturbation.

Multiplicative perturbation: Two basic forms of multiplicative noise have been studied in the Statistics community [13]. One multiplies each data element

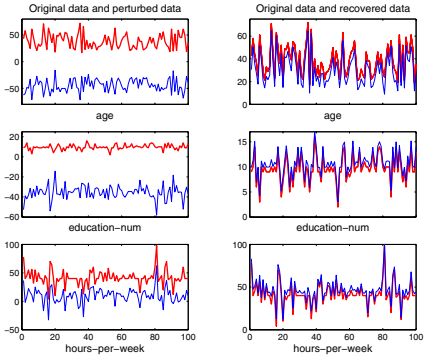


Fig. 3. (Left Column) The first 100 records from the original data and the perturbed data. (5% samples) (Right Column) The first 100 records from the original data and the recovered data. (5% samples).

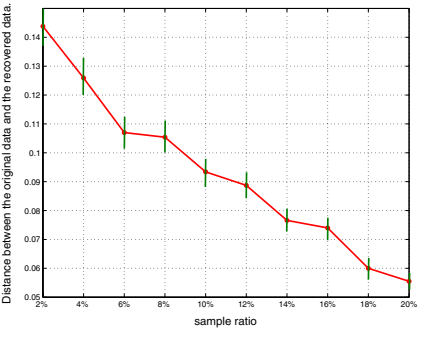


Fig. 4. Performance (average of 20 independent trials) of PCA-based attack w.r.t. sample size for Adult data. Error bars show 95% confidence intervals.

by a random number that has a truncated Gaussian distribution with mean one and small variance. The other takes a logarithmic transformation of the data first, adds multivariate Gaussian noise, then takes the exponential function *exp(.)* of the noise-added data. Neither of these perturbations preserve distance and are fundamentally different than the type we study, orthogonal transformations. To facilitate large scale data mining applications, Liu *et al.* [14] proposed an approach where the data is multiplied by a randomly generated matrix – in effect, the data is projected into a lower dimensional space. This technique preserves distance on expectation. However, the privacy analysis there did not take into account prior knowledge as we do. Oliveira and Zaiane [6], Chen and Liu [5] discuss the use of random rotation for privacy-preserving clustering and classification. These authors observe that the distance preserving nature of random rotation makes it useful in this setting, but do not analyze its privacy limitations.

Categorical data perturbation: Evfimievski *et al.* [15], Rizvi and Haritza [16] consider the use of data categorical perturbation. They develop algorithms from which association rules present in the original data can be estimated from the perturbed data. Along a related line, Verykios [18] consider perturbation techniques which allow the discovery of *some* association rules while hiding others considered to be sensitive.

Data anonymization: Sweeney [4] developed the *k-anonymity* framework wherein the original data is transformed so that the information for any individual cannot be distinguished from k-1 others. Values from the original data are generalized (replaced by a less specific value) to produce the anonymized data. This technique makes no accuracy guarantees for subsequent analysis of the transformed data.

Data swapping: This technique transforms the database by switching a subset of attributes between selected pairs of records so that the individual record entries are unmatched, but the statistics are maintained across the individual fields. A variety of refinements and applications of data swapping have been addressed since its initial appearance. We refer readers to [19] for a thorough treatment.

6 Conclusions

We considered the use of distance-preserving maps as a data perturbation technique for privacy-preserving data mining. On the one hand, this technique is quite useful as it is computationally efficient, and it allows many interesting data mining algorithms to be applied directly to the perturbed data and produce an error-free result *e.g.* K-means clustering and k-nearest neighbor classification. On the other hand, the privacy offered by distance preserving transformations has, to our knowledge, not been well-studied. We take a step in this direction by considering two types of prior knowledge an attacker may have and use to design attack techniques to recover the original data. The first is based on basic properties of linear algebra and the second on principal component analysis.

We conclude the paper by pointing out a potential remedy to the privacy problems described earlier for the PCA attack. Recall that the attacker, with a good estimate of the original and transformed covariance matrices, could gain a lot of information about the orthogonal transformation T itself and, therefore, undo it quite well to recover the original data. We suggest, however, that the data owner instead use a randomized transformation which is orthogonal on expectation – namely, random projection. The owner generates $\hat{R}$, a $\ell \times n$ matrix with each entry sampled independently from a distribution with mean zero and variance one and releases $Y = RX$ where $R = \ell^{-1/2}\hat{R}$ (this type of data perturbation for $\ell \leq n$ was discussed in [14]). It can be shown that matrix R is orthogonal on expectation and the probability of orthogonality approaches one exponentially fast with ℓ. By increasing ℓ, the data owner can guarantee that distances are preserved with arbitrarily high probability. However, it can be shown that the randomness introduced by R kills the covariance in Y used by the PCA based attack. Specifically, given random vector V, it can be shown that, Σ_{RV} (the covariance matrix of RV) equals $I_n\gamma$ for some constant γ. *Any* vector in $\mathbb{R}^n$ is an eigenvector of Σ_{RV}, therefore, the PCA based attack will not work. The exploration of this kind of randomized orthogonal transformation is a good direction for future work.

Acknowledgment

This research is supported by the U.S. NSF Grant IIS-0329143. H. Kargupta also received partial support from NSF CAREER award IIS-0093353. The authors thank R. Wolff for several helpful discussions.

References

1. Agrawal, R., Srikant, R.: Privacy preserving data mining. In: Proc. ACM SIGMOD. (2000) 439–450
2. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: Random data perturbation techniques and privacy preserving data mining. Knowledge and Information Systems **7**(5) (2005) 387–414
3. Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In: Proc. ACM SIGMOD. (2005) 37–48
4. Sweeney, L.: K-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems **10**(5) (2002) 557–570
5. Chen, K., Liu, L.: Privacy preserving data classification with rotation perturbation. In: Proc. IEEE ICDM. (2005) 589–592
6. Oliveira, S.R.M., Zaïane, O.R.: Privacy preservation when sharing data for clustering. In: Proc. Workshop on Secure Data Management in a Connected World. (2004) 67–82
7. Artin, M.: Algebra. Prentice Hall (1991)
8. N. R. Adam, J.C.W.: Security-control methods for statistical databases: A comparative study. ACM Computing Surveys **21**(4) (1989) 515–556
9. Jolliffe, I.T.: Principal Component Analysis. Second edn. Springer Series in Statistics. Springer (2002)
10. G. Strang: Linear Algebra and Its Applications (3rd Ed.). Harcourt Brace Jovanovich College Publishers, New York (1986)
11. Székely, G.J., Rizzo, M.L.: Testing for equal distributions in high dimensions. InterStat **November**(5) (2004)
12. Vaidya, J., Clifton, C., Zhu, M.: Privacy Preserving Data Mining. Volume 19 of Series: Advances in Information Security. Springer (2006)
13. Kim, J.J., Winkler, W.E.: Multiplicative noise for masking continuous data. Technical Report Statistics #2003-01, Statistical Research Division, U.S. Bureau of the Census (2003)
14. Liu, K., Kargupta, H., Ryan, J.: Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining. IEEE Transactions on Knowledge and Data Engineering **18**(1) (2006) 92–106
15. Evfimevski, A., Gehrke, J., Srikant, R.: Limiting privacy breaches in privacy preserving data mining. In: Proc. ACM PODS. (2003)
16. Rizvi, S.J., Haritsa, J.R.: Maintaining data privacy in association rule mining. In: Proc. 28th VLDB. (2002) 682–693
17. Hore, B., Mehrotra S., Tsudik G.: A privacy-preserving index for range queries. In: Proc. 30th VLDB. (2004) 720–731
18. Verykios, V.S., Elmagarmid, A.K., Elisa, B., Saygin, Y., Elena, D.: Association rule hiding. IEEE Transactions on Knowledge and Data Engineering **16**(4) (2004) 434–447
19. Fienberg, S.E., McIntyre, J.: Data swapping: Variations on a theme by dalenius and reiss. Technical report, U.S. National Institute of Statistical Sciences (2003)

A Scalable Distributed Stream Mining System for Highway Traffic Data*

Ying Liu[1], Alok Choudhary[2], Jianhong Zhou[3], and Ashfaq Khokhar[3]

[1] Graduate University of Chinese Academy of Sciences
Data Technology and Knowledge Economy Research Center, Chinese Academy of Sciences
Beijing, China 100080
yingliu@gucas.ac.cn
[2] Electrical and Computer Engineering Department, Northwestern University
Evanston, IL, USA 60208
choudhar@ece.northwestern.edu
[3] Computer Science Department, University of Illinois at Chicago
Chicago, IL, USA 60607
{jzhou13, ashfaq}@uic.edu

Abstract. To achieve the concept of smart roads, intelligent sensors are being placed on the roadways to collect real-time traffic streams. Traditional method is not a real-time response, and incurs high communication and storage costs. Existing distributed stream mining algorithms do not consider the resource limitation on the lightweight devices such as sensors. In this paper, we propose a distributed traffic stream mining system. The central server performs various data mining tasks only in the training and updating stage and sends the interesting patterns to the sensors. The sensors monitor and predict the coming traffic or raise alarms independently by comparing with the patterns observed in the historical streams. The sensors provide real-time response with less wireless communication and small resource requirement, and the computation burden on the central server is reduced. We evaluate our system on the real highway traffic streams in the GCM Transportation Corridor in Chicagoland.

Keywords: data stream, distributed computing, real-time, traffic, sensor.

1 Introduction

Advances in computing and communication over wired and wireless networks have resulted in many pervasive distributed computing environments, such as PDAs, cell phones, sensors, etc. Data in such applications is increasingly getting transformed to continuous data streams, which are dynamically changing, exhibit high volume, have high dimensionality, and are potentially infinite in length. Several issues related to mining stream data have been investigated in [1, 2, 3, 4, 8, 11]. The ever-increasing computational capacity of ubiquitous equipments presents an opportunity for intelligent data analysis to be performed "anytime, anywhere" with constrained resources. Stream mining on pervasive computing devices has a broad range of applications. For example, commercial fleet management companies spend a lot of

* This work was done during the first author's doctoral study in Northwestern University.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 309 – 321, 2006.
© Springer-Verlag Berlin Heidelberg 2006

time and labor in collecting vehicle performance data, studying the data offline, and estimating the condition of the vehicle primarily through manual efforts. If on-board PDA was installed in every vehicle and connected with the remote server through wireless networks, real-time analysis of vehicle data streams would be achieved by the PDAs with less communication with the server. Patient health monitoring, forest fire monitoring, and security surveillance by sensor networks are also good examples.

Mining of highway traffic data, such as that of Gary-Chicago-Milwaukee (GCM) Corridor Transportation System, is a typical example of distributed stream mining application, where hundreds of sensors collect and send the traffic status data to a central server all day long over the expensive wireless and wired connections. Currently, a central server collects all the data and performs congestion analysis offline. This traffic analysis results are then published to the travelers through a central interface. However, this centralized process has several shortcomings: 1) over 1.2 GB data is sent to the server every day (5 MB per sampling) in GCM, which must be a communication burden on low bandwidth networks. 2) The huge amount of data per day is also a heavy burden on the storage system. 3) The traffic information provided to the travelers is not a real-time response due to the communication overhead. 4) This traffic data corresponds to only a small fraction of roads in the Chicago area and thus is likely to increase over time when more roads are equipped with sensors. There is an astonishing fact that traffic congestion wastes 2 billion gallons of fuel per year in the United States alone. 135 million US drivers spend 2 billion hours trapped in congestion per year. The total cost to Americans due to traffic congestion exceeds $100 billion per year. It would be highly beneficial to predict the congestion level as early as possible so that the drivers can avoid being trapped by choosing another route in advance. Therefore, there is an urgent demand for a traffic analysis system where traffic data is processed and mined in a distributed fashion, and the sensors are able to perform stream data analysis techniques on the fly (such as abnormal events real-time detection, traffic jam prediction, flow speed prediction, etc.). This demand for such a system is likely to increase with the increase in the use of mobile database devices inside the vehicles. As a matter of fact, this problem is very difficult because of the following issues:

1) Very little research has been done in distributed stream mining. Most of the existing algorithms are designed to work as a centralized application. This type of approaches is not scalable when the number of ubiquitous devices is large and cannot provide real-time response.
2) Although some distributed data mining algorithms have been proposed in [10], they don't consider the unique characteristics of data streams that the patterns change dynamically.
3) Sensors are lightweight devices, short of power supply, computation capability and storage size. Data mining on resource-constrained devices is not well explored yet.

In this paper, we propose a scalable distributed traffic stream mining system. It is designed to monitor the current roadway traffic status and predict the coming traffic in real-time. The proposed system consists of the following four phases: preprocessing, training, monitoring/predicting and updating. In the training phase, the central server performs various offline data mining techniques on the cleaned historical data streams, and ships the discovered patterns back to the sensors. Based

on the similarity of the patterns, it also groups the sensors into clusters. In the monitoring/predicting phase, each sensor predicts the coming traffic using the global patterns. If an "abnormal" event is predicted or observed at a sensor, the sensor raises an alarm immediately. The alarm is sent to the central server which can notify all the members in the group. Real-time analysis is achieved because each sensor works independently and incurs no communication delay. The updating phase is triggered periodically or when the number of mispredictions exceeds a threshold. There are three main characteristics of our real-time distributed traffic stream mining system: 1) The central server takes the major computation tasks in the training phase and the updating phase using its strong computation capability; 2) the sensors or similar lightweight devices perform predicting or monitoring using their constrained resources; 3) the data mining techniques used in the updating phase update the patterns efficiently. The design of this system tries to target the three challenges mentioned above. The proposed framework can be applied to similar applications, like forest fire monitoring, vehicle health monitoring, etc.

Data from a real roadway transportation system (GCM) is used in our experiments to evaluate our proposed system. It shows good scalability in various aspects. Memory usage on each sensor and communication overheads are small.

The rest of this paper is organized as follows. Section 2 briefly introduces the related algorithms and systems. Section 3 describes a highway transportation system. Section 4 presents the overall architecture of the real-time distributed traffic stream mining system and describes the methodology in each phase, respectively. Experimental results are presented in Section 6, and Section 7 summarizes this paper.

2 Related Work

2.1 Stream Mining

Lossy Counting [8] presents an algorithm for computing frequency counts exceeding a user-specified threshold over data streams. Gianella et al. [3] proposed an efficient approach to mine time-sensitive frequent patterns. It incrementally maintains only the historical information of (sub)frequent patterns. In order to adapt quickly to the changes of the underlying data stream, Aggarwal et al. [2] trains and tests streams simultaneously by selecting an appropriate window of past data to build the classifier. In [4] Guha et al. proposed a constant-factor approximation algorithm for K-Median problem in data stream clustering. This algorithm maintains a consistently good quality using a small amount of memory and time.

2.2 Time-Series Data Mining

A time-series database consists of sequences of values or events changing with time. Similarity search [12] finds sequences that differ only slightly from a given sequence. Similarity search analysis is useful in stock data analysis, cardiogram analysis, traffic patterns, power consumption analysis, etc.

2.3 Distributed Stream Mining

As many large transaction databases are available, [15] proposes Fast Distributed Mining (FDM), which generates a small number of candidate sets and reduces the number of messages to be passed. [14] extends the traditional ARM to peer-to-peer computing environments. This algorithm combines ARM, executed locally at each node, with a majority voting protocol to discover all the rules that exist in the combined database. It is asynchronous and communication efficient.

VEDAS, a real-time on-board stream mining system for vehicle-health-monitoring and driver status characterization, is developed in [6]. The PDAs perform most of the data management and mining tasks and send the analysis results to the central server site. This system minimizes bandwidth usage by limiting the centralization of the sensed data. The mobile stock monitoring system in [7], and the health monitoring tool in [5] are similar to VEDAS in terms of conceptual design. Our proposed system is sensor networks based, where the sensors have very limited computation capability and resources. Therefore, the central server has to take the main computation tasks.

2.4 Roadway Traffic Management System

Contemporary roadway traffic management systems have investigated issues related to route planning, automatic accident detection, short-term travel prediction, and better user interfaces for answering these questions. Advanced traffic management systems such as TrafficWise (Indiana's Intelligent Transportation System) rely on traffic flow speed sensors and cameras, as well as information from emergency responders and roadside assistance patrols, to detect traffic problems and determine the underlying causes. The information is centrally collected, analyzed, and then the results are delivered back to drivers, dispatchers, and emergency responders. Grossman et al. [13] have developed a framework that detects real-time changes in highway traffic data and send real-time alerts. However, the data collected at different sensors are centralized to a supercomputer cluster system for analysis. In contrast, our system doesn't need to transmit any data streams unless updating the patterns.

3 Gary-Chicago-Milwaukee (GCM) Corridor Transportation Data

The GCM Corridor (Figure 1) consists of the 16 urbanized counties and 2,500 miles of roadways which connect the three cities. 855 sensors are placed on the roads, and each sensor collects 288 streams every day (one sampling every 5 minutes). Each sensor collects the real-time traffic data at its location and sends it to the central server over wireless connections periodically. Each stream consists of static attributes (longitude, latitude, length, direction, etc.) and dynamic attributes (vehicle speed, congestion level, occupancy, volume, etc.)

GCM travel system is providing a number of services to travelers, for example, in Figure 1, the different colors on the roadways mean different congestion levels. Currently, all of the data analysis is still performed offline on a central server. It is not real-time response because the sensors collect data every 5 minutes. In addition, the computation burden on the server is heavy.

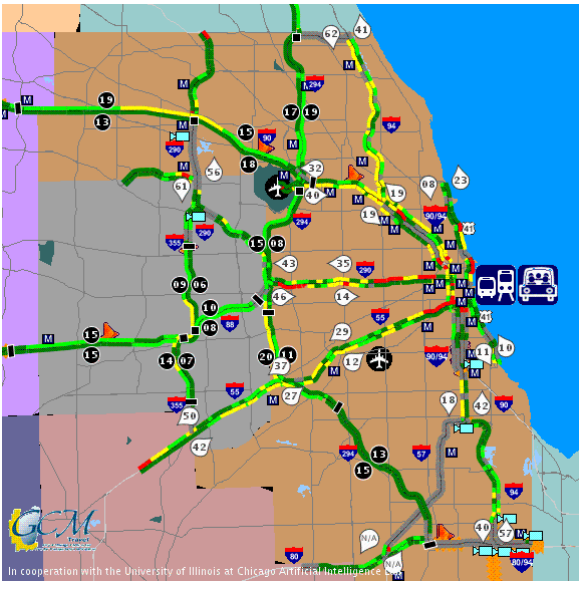


Fig. 1. Gary-Chicago-Milwaukee Corridor Transportation System

4 Framework of Distributed Traffic Stream Mining System

In order to reduce data transmission and computation time of the traditional centralized stream mining model, we propose a distributed traffic stream mining system. In our proposed system, the central server performs data mining techniques to discover patterns in the training phase and update patterns in the updating phase, respectively. The sensors perform monitoring and predicting tasks. This system incurs little data transmission except the transmission for the discovered patterns, alerts and the new streams for rebuilding the model. As the sensors in a roadway traffic system usually do not have as sufficient power supply or computation capability as on-board PDAs in VEDAS, most of the computation has to be assigned to the central server. Figure 2 shows the framework of our proposed system, which consists of four phases: *preprocessing, training, monitoring/predicting, and updating*. Each phase is described in detail in the following subsections.

5.1 Preprocessing Phase

The central server collects streams from the distributed servers, then, extracts time-series attributes that dynamically change with time (vehicle speed, congestion level, etc.) from the raw data, and eliminates the static attributes. Clean the outliers and fill up the missing numbers.

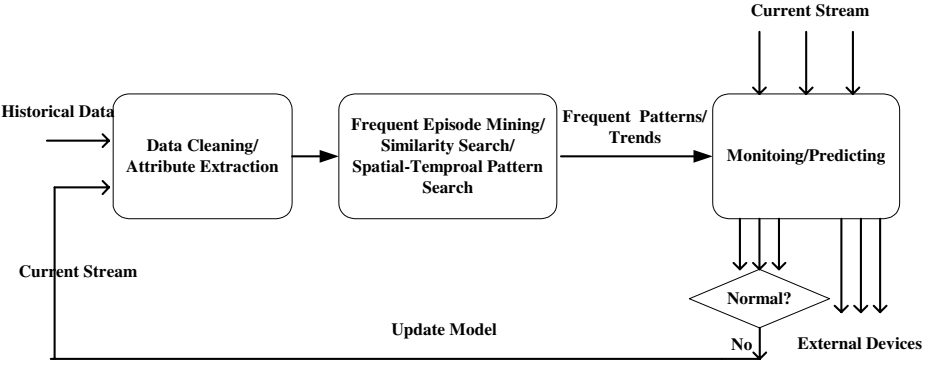


Fig. 2. Framework of the distributed traffic stream mining system

5.2 Training Phase

In this phase, the central server performs various data mining techniques on the preprocessed data. A number of interesting patterns or trends can be mined. Then, the interesting patterns are relayed to the sensors over the wireless network connections. Since the congestion level prediction is very valuable to travelers as mentioned in Section 1, we use "frequent episode mining" technique, which can help predict the congestion level, as an example to illustrate the training phase. We would like to emphasize that there are several algorithms can be applied to this application. However, due to limitations in space, we only present the details of one technique, although we would like to provide more.

Frequent Episode Mining. In roadway traffic data, the "congestion level" observed by each sensor can be viewed as a sequence (A sequence is a series of events where each event has an associated timing stamp.). A congestion level sequence database is created by putting all the sequences from all the distributed sensors together. An example of a congestion level sequence taking place at a sensor is

(non, 06:05), (light, 06:10), (medium, 06:15), (light, 06:20), (medium, 06:25), (heavy, 06:30)

The numbers indicate the timings when the events happened. The episode "light_congestion followed by medium_congestion" occurs twice in a window of 6 events.

Frequent episode mining is to find the collections of events which occur so frequently that exceed a pre-defined threshold [9] in the sequence database. Note that, here, we are only interested in the episodes where the events have no intervals in between. In the example sequence, if the threshold is set at 2, "non_congestion followed by medium_congestion" is not considered as a frequent episode because "medium_congestion" doesn't follow "non_congestion" immediately. Figure 3 presents the pseudo code of the frequent episode mining algorithm used in our system. It adopts an Apriori-like level-wise searching approach that generates new candidates and calculates their supports by scanning the database at each level. Remember that all the subepisodes of any frequent episode must be frequent. Please note line 6 in Figure 3. For each occurrence of candidate episode c in sequence s, the support of c is increased by 1, which is different from the support counting in association rule mining

```
Input: sequence database SD, congestion level set E, min support threshold min_fr
Output: frequent episodes collection FE
1.  C_1 := {{e} | e ∈ E};
2.  i := 1;
3.  while C_i ≠ ∅ do
4.    forall s ∈ SD do          //scan database, calculate support
5.      forall c ∈ C_i do
6.        for each occurrence of c in s, c.support ++;
7.      end forall
8.    end forall
9.    forall c ∈ C_i do      //identify frequent episodes
10.     if c.support > min_fr
11.       add c to FE_i;
12.     end if
13.   end forall
14.   build C_{i+1} from FE_i;//generate candidate episodes
15.   i := i + 1;
16. end while
17. FE := ∪ FE_i;
```

Fig. 3. Pseudo code of the frequent episode mining algorithm

where the support of c is only increased by 1 no matter how many times c occurs in s. In addition, we keep track of which sensor and when each frequent episode happens.

The frequent congestion level episodes can help transportation domain experts answer questions such as how the episodes evolve over time by sorting the interesting patterns by the timing stamps, how a congestion propagates spatially (what is the origin of the congestion and by what routes the congestion propagates) by combining the knowledge of the spatial relations (distance, direction, inter-connection, etc.) between the corresponding sensors, etc. Spatial effects are indirectly taken into account because we do observe unusual effects of traffic pattern in one area to another one in a "seemingly unrelated area", which is not so obvious.

Consistent Pattern. We proceed to find the *consistent patterns*. Assume we have N days' data and a user-specified threshold M. If a frequent episode *fe* happens on M out of the N days at a sensor A at a certain time t, we call (A, t, *fe*) a *consistent pattern*. All the sensors that have a common *fe* are clustered into a same group. A counter is maintained in each group to record the number of updating request. Finally, we send the consistent patterns to the corresponding sensors over wireless connections. We are interested in the consistent patterns and the corresponding sensors because the patterns tend to re-occur. In addition, since the sensors in the same group show similar traffic flow trends for known or unknown reason, a sensor's coming traffic could be predicted based on the most recent observations from its group members.

One may have a question that why not to let each sensor predict its coming traffic by using its local statistical values, such as the mean speed of the traffic flow

observed in the past few days at a given time, or the mean congestion level in the past few days at a given time, or the observations of the last day. The answer is in the following three aspects: 1) the storage cost is higher than using our proposed framework and algorithms. A sensor needs a relatively large memory to store the streams in the past few days in order to maintain the statistical numbers. In contrast, in our system, each sensor only stores the frequent patterns delivered by the central server. 2) It requires higher computation capability and power consumption for a sensor to calculate the statistical values, which may exceed the capability of any similar lightweight devices in pervasive environments. In contrast, the computation on each sensor in our system is very simple: Compare the new observations with the patterns; if not matching, send an updating request to increase its group counter by 1, otherwise, do nothing. 3) Using local statistics, a sensor only sees its local view with no chance to know the global view of the neighboring areas, which actually may impact its traffic flow. A group of sensors that share a common *consistent pattern* may have similar traffic behaviors. Therefore, it may be more confident to predict the coming traffic at a certain sensor based on an event happened a moment ago. For example, consider that sensors *A, B,* and *C* always experience the same congestion level sequence (*light, medium, medium, medium*) from *6:30am*, *6:50am*, and *7:20am* for the subsequent 20 minutes, respectively. *A, B,* and *C* may or may not be spatially connected to each other. If today, somehow, the congestion level sequence observed at *A* at *6:30am* is (*medium, heavy, heavy, heavy*), it is highly likely that this early rush hour jam will happen at *B* and *C* soon. So *A* will send a warning to the server, and then the server will notify *B* and *C* so that they can send out "early traffic jam" warning signals to its local external devices in advance.

Predictive models (decision tree, logistic regression, Naive Bayesian classifier, etc.) could be used to predict if a heavy congestion would happen in the next moment. However, either the computation or the storage cost (there must be a number of rules for different time slots) is larger than those of the frequent episode mining algorithm.

5.3 Monitoring/Predicting Phase

Each sensor stores its significant patterns mined from the recent historical data, such as the frequent congestion level episodes at some timing points, the speed fluctuating patterns, etc. The monitoring/predicting phase is designed to predict the local roadway coming traffic flow and detect abnormal events deviating from the observed patterns. For example, sensor *X* predicts the forthcoming congestion level *l*. If *l* is a severe situation, then *X* will send a warning to its external device and an alarm to the server as well. Then, the server will notify all the members of the cluster containing *X*. If *l* is different from the actual level *l'*, sensor *X* will send an updating request to augment the counter in its corresponding group on the server. The server maintains a counter for each group. The aim of the predictions is to provide travelers with up-to-date information so that they can choose routes to avoid being stuck in traffic jams.

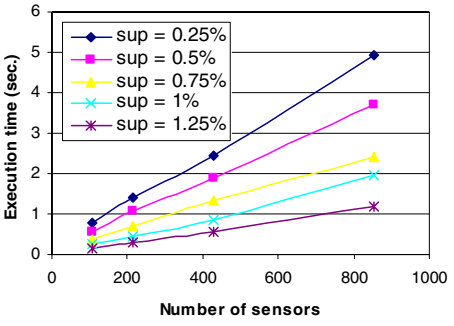


Fig. 4. Execution time on one day's data with varying number of sensors. The number of sensors varied from 107 to 855.

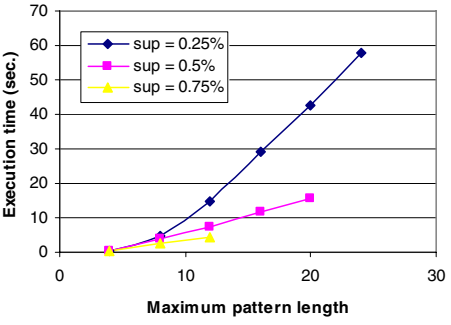


Fig. 5. Execution time on one day's data. The maximum pattern length varied from 4 to 24.

5.4 Updating Phase

While monitoring their local traffic streams, the sensors send updating requests to augment the counter in its corresponding group on the server whenever any misprediction happens. The server maintains a counter for each group. Once the counter exceeds a user-specified threshold, the server starts to download new streams from the sensors which send out the requests. Thus, the data transmission cost is much smaller than that of collecting new streams from all the sensors. It is not a heavy burden for the central server, and will not exhaust the sensors' power.

One important feature of data stream is that its patterns change dynamically, therefore, there is no need to keep the old patterns when updating the patterns. The server replaces the oldest day's data in the database with today's new data. Then, it starts to perform some corresponding data mining algorithms. Finally, the updated patterns are relayed to the sensors over wireless connections. The members in this specific group may have to wait for one day to be informed the latest patterns.

6 Experimental Results

We evaluate our distributed traffic stream mining system on real traffic streams from a roadway transportation system, Gary-Chicago-Milwaukee (GCM) Corridor Transportation System (See details of GCM in Section 3). For the purpose of our experiment, we extract the dynamic attribute "congestion level". Each sensor of the 855 sensors contributes a sequence of 288 congestion levels every day. There are four different congestion levels: *non, light, medium, and heavy*. We download the traffic streams of 5 weekdays.

In this section, we only discuss the performance of frequent episode mining. Its scalability and prediction accuracy are analyzed. Communication overheads and memory usage are also discussed. All the experiments are performed on a 700-MHz

Table 1. Traffic congestion level prediction accuracy

Days for training	Threshold	Days for predicting	Avg. cluster size	Accuracy
day_1,day_2,day_3	3	day_4	10.3	25.2%
day_1,day_2,day_3	3	day_5	10.3	7%
day_2,day_3,day_4	3	day_5	22.4	11.9%
day_1,day_2	2	day_3	106.4	9.7%
day_1,day_2	2	day_4	106.4	7.7%
day_1,day_2	2	day_5	106.4	8.6%
day_2,day_3	2	day_4	105.8	21.5%
day_2,day_3	2	day_5	105.8	8.9%
day_3, day_4	2	day_5	291.8	6.6%

Xeon 8-way shared memory parallel machine with a 4GB memory, running the Red Hat Linux Advanced Server 2.1 operating system. The program is implemented in C.

6.1 Scalability in Training Phase

Since the central server performs the data mining tasks in the training stage, we would like to investigate the scalability of the data mining algorithms on the server, specifically the scalability of frequent episode mining when varying the number of sensors, minimum support or maximum pattern length.

As the number of sensors in different systems may be different, the scalability when varying the number of sensors is evaluated in Figure 4. The number of sensors is varied from 107 to 855. The maximum length of a frequent episode is set to be 8. The execution time on one day's (24 hours) congestion level sequences scales well when the number of sensors is increasing.

In order to provide real-time congestion prediction, short episode is more useful. Thus, we restrict the maximum length of the frequent episodes in each run. The total number of frequent episodes increases as the maximum length increases. Figure 5 presents the execution time on one day's (24 hours) congestion level sequences. The support threshold is varied from 0.25% to 0.75%. When the support is set at 0.25%, the execution time is increasing fast because both the number of frequent episode candidates and the number of database scans are huge due to the low support threshold. For the case of 0.5% and 0.75%, it scales well as the length of episodes increases.

6.2 Traffic Prediction

As described in Section 5.2, all the sensors (associated with the same timing *t*) that have a common "consistent frequent episodes *fe*" are clustered into the same group. Then the consistent frequent episodes are sent to the corresponding sensors over wireless connections for monitoring or predicting purpose. We use a fraction of the 5 weekdays' data for training and the rest for predicting. Table 1 shows the prediction

Table 2. Average number of patterns on each sensor

Days for training	Consistent pattern threshold	Avg. # of patterns / sensor
day_1, day_2, day_3	3	0.46
day_2, day_3, day_4	3	0.46
day_3, day_4, day_5	3	0.97
day_1,day_2	2	4.6
day_2, day_3	2	4.6
day_3,day_4	2	14

accuracy. The maximum pattern length is set to be 8 because users are more interested in short-term predictions. The support threshold is 0.75%. Overall speaking, the prediction accuracy is not high. The best case happens when using day_1, day_2 and day_3's patterns to predict day_4's traffic, where the overall accuracy is 25.2%. In addition, in this case, 27% of the sensors get 100% prediction accuracy (All the predictions made at these sensors match the actual streams.). The results in this table verify the dynamic nature of traffic data streams.

As described in the monitoring/predicting phase in Section 5.3, if an "abnormal" event is observed, the sensor sends an alert to the central server, and then, the server notifies all its group members. In our experiments, we observe that the warnings are pretty helpful in some cases. For example, sensor #337, #415, #731 and #732 are in the same group because they have (*light, light, light*) at *1:00am, 1:20am, 3:10am,* respectively. The congestion levels on sensor #337 at *1:00am* was (*medium, medium, medium*), deviating from the pattern (*light, light, light*). Then, (*medium, light, medium*) happened on sensor #415 at *1:20am*, (*medium, medium, light*) happened on sensor #731 and #732 at *3:10am*. If the server sent an "abnormal congestion" alert to sensor #415, #731 and #732 at *1:15am*, they would have sent warning signals to the drivers in advance, so that they could avoid the jam near sensor #731 and #732 at *3:10am*. This observation verifies our claim that congestions may propagate spatially and temporally between roadways where similar behaviors often happen.

6.3 Storage

As the storage resource on a sensor or a lightweight computing device is very limited, we would like to investigate the cost for storing the patterns. Table 2 presents the average number of patterns stored on each sensor. We use 3 days', and 4 days' data for training and set the "consistent" pattern threshold as 2, 3, respectively. From Table 2 we can see that the average number of patterns on each sensor is quite small for any case. Therefore, our proposed system doesn't require a large storage resource.

6.4 Communication Cost

The number of mispredicitons is small, ranging from 300 to 3000 per day. Thus, the data transmission cost must be small, because there is no data transmission between

the central server and a sensor unless a misprediction happens. In contrast, if the central server were to download all the streams from all the sensors to analyze and then provide up-to-date information, the data transmission cost must be huge.

7 Conclusions and Open Issues

This paper presented a scalable distributed traffic stream mining system. In the training phase, the central server performs various data mining techniques on historical data streams to find useful patterns. In the monitoring/predicting phase, each sensor predicts the forthcoming data or raises alarms if an abnormal situation is predicted by comparing with the patterns in recent historical streams. Our experiment results on the real traffic data from GCM demonstrate that this model is scalable and effective. The "abnormal" congestion warning is helpful. This system can provide travelers with real-time traffic information and help transportation domain experts understand the characteristics of a transportation system. It achieves real-time analysis with low communication cost. In addition, the computation resource requirement on each sensor is small as well as the storage requirement.

The proposed system is at an early stage of development and will be substantially enhanced by incorporating the following aspects:

1) The most challenging part is how to build a model to involve both temporal and spatial features. We plan to use connected graph to represent the spatial relations between different sensors, and incorporate graph mining algorithms to our system. The prediction accuracy must be improved a lot by then.
2) In order to discover more interesting patterns, more stream mining techniques applicable to traffic streams should be explored.
3) Building of a simulation environment where we can see how the "early" warnings affect the traffic flow.

Acknowledgments. This work was supported in part by National Science Foundation grants IIS-0536994, CNS-0551639, CNS-0406341, Intel, IBM, CNS-0550210, and in part by National Natural Science Foundation of China Project #70531040, #70472074, Ministry of Science and Technology of China 973 Project #2004CB720103.

References

1. Aggarwal, C., Han, J., Wang, J., and Yu, P. S.: A Framework for Clustering Evolving Data Streams. In Proc. Intl. Conf. on Very Large Data Bases (VLDB), 2003
2. Aggarwal, C., Han, J., Wang, J., and Yu, P. S.: On Demand Classification of Data Streams. In Proc. 2004 Intl. Conf. on Knowledge Discovery and Data Mining (KDD), 2004
3. Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P. S.: Mining Frequent Patterns in Data Streams at Multiple Time Granularities. H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining, 2003
4. Guha, S., Mishra, N., Motwani, R., and O'Callaghan, L.: Clustering Data Streams. In Proc. IEEE FOCS Conf., 2000

5. Jorge, A.: Adaptive Tools for the Elderly: New Devices to Cope with Age-induced Cognitive Disabilities. In Proc. EC/NSF workshop on the Universal accessibility of ubiquitous computing: providing for the elderly, 2001, 66-70
6. Kargupta, H., Bhargava, R., Liu, K., Powers, M., Blair, P., Bushra, S., Dull, J., Sarkar, K., Klein, M., Vasa, M., and Handy, D.: VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. In Proc. SIAM International Conf. on Data Mining, 2004
7. Kargupta, H., Park, B. H., Pittie, S., Liu, L., Kushraj, D., and Sarkar, K.: MobiMine: Monitoring the Stock Market from a PDA. SIGKDD Explorations, Jan. 2002, 3(2): 37-46
8. Manku, G. S., and Motawani, R.: Approximate Frequency Counts over Data Streams. In Proc. 28th Intl. Conf. on Very Large Databases (VLDB), 2002
9. Mannila, H., Toivonen, H., and Verkamo, A. I.: Discovering frequent episodes in sequences, In Proc. Intl. Conf. on Knowledge Discovery and Data Mining, 1995
10. Park, B., and Kargupta, H.: Distributed Data Mining: Algorithms, Systems, and Applications. Data Mining Handbook, 2002
11. Toivonnen, H.: Sampling large database for association rules. In Proc. 22nd Intl. Conf. on Very Large Databases, 1996, 134–145
12. Agrawal, R., Faloutsos, C., and Swami, A.: Efficient Similarity Search in sequence databases. In Proc. 4th Intl. Conf. Foundations of Data organization and Algorithms, 1993
13. Grossman R., Sabala, M., Alimohideen, J., Aanand, A., Chaves, J., Dillenburg, J., Eick, S., Leigh J., Nelson, P., Papka, M., Rorem, D., Stevens, R., Vejcik, S., Wilkinson, L., and Zhang, P.: Real Time Change Detection and Alerts from Highway Traffic Data, In Proc. Intl. Conf. Super Computing, 2005
14. Wolff, R., and Schuster, A.: Association Rule Mining in Peer-to-Peer systems. ICDM, 2003
15. Cheung, D., Han, J., Ng V., Fu, A. W., Fu, Y.: A Fast Distributed Algorithm for Mining Association Rules. In Proc. of PDIS, 1996

K-Landmarks: Distributed Dimensionality Reduction for Clustering Quality Maintenance*

Panagis Magdalinos[1], Christos Doulkeridis[1], and Michalis Vazirgiannis[1,2,**]

[1]Athens University of Economics and Business
Patision 76, 10434, Athens, Greece
[2]INRIA FUTURS
Parc Club Orsay Universite 91893, France
{pmagdal, cdoulk, mvazirg}@aueb.gr

Abstract. Due to the vast amount and pace of high-dimensional data production and their distribution among network nodes, the fields of Distributed Knowledge Discovery (DKD) and Distributed Dimensionality Reduction (DDR) have emerged as a necessity in many application areas. While a wealth of centralized dimensionality reduction (DR) algorithms is available, only few have been proposed for distributed environments, most of them adaptations of centralized ones. In this paper, we introduce K-Landmarks, a new DDR algorithm, and we evaluate its comparative performance against a set of well known distributed and centralized DR algorithms. We primarily focus on each algorithm's performance in maintaining clustering quality throughout the projection, while retaining low stress values. Our algorithm outperforms most other algorithms, showing its suitability for highly distributed environments.

Keywords: Distributed dimension reduction, distributed knowledge discovery.

1 Introduction

Distributed Knowledge Discovery (DKD) has emerged as one of the most challenging tasks in large scale distributed data management. This is partly due to the inapplicability of centralized approaches in current research problems, which is more evident with the advent of new application areas that are inherently distributed, such as sensor networks and peer-to-peer (P2P) systems. The main characteristic of P2P systems is the lack of global knowledge, in the sense that no peer can gather all available data. In large scale P2P networks, data is distributed to peers (in horizontal partitioning manner) making the cost of centralized assembly and subsequent computation of any centralized algorithm prohibitive. On

* This work is co-funded by the European Social Fund (75%) and National Resources (25%) - (EPEAEK II) - PYTHAGORAS.

** The work of Dr. Vazirgiannis is supported by the EU Marie Curie Intra-European Fellowship.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 322–334, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the other hand, globally described data can be of very high dimensionality, while peer local dimensions can be different from each other (vertical partitioning).

Dimensionality reduction techniques tackle these problems through the definition of various methods for embedding the data from the initial space R^n to the target space R^k, where $k < n$. These algorithms are extremely useful in various disciplines related to knowledge discovery. The latter becomes a difficult task as the number of dimensions increases, because of two distinct problems: the "empty space phenomenon", and the "curse of dimensionality" [4]. The former denotes the fact that data in high dimensional spaces is sparsely situated, having almost equal distance from one another. The latter refers to the fact that the sample needed to estimate a function of several variables to a given degree of accuracy grows exponentially with the number of variables. A thorough investigation of both problems can be found in [5].

The motivation for our work emerges from the need to apply dimensionality reduction on data distributed in a P2P network. This task is directly applicable in P2P information retrieval applications, where documents are represented as high dimensional points using the vector space model. Distributed dimensionality reduction (DDR) algorithms are then necessary to decrease the representation costs and to reveal potentially interesting or hidden structure in the data.

Towards this objective, we focus on the DKD problem, assuming that the data set is partitioned horizontally (i.e. non overlapping sets of identically structured tuples) and distributed on peers. We identify the following requirements for DDR algorithms: 1) each point's projection should be computed independently from other points, 2) distances between points should be preserved, 3) the algorithm should be fast and linear to the number of projected points, and 4) the algorithm should incur low communication cost (in a distributed context).

In this paper, we present a DDR algorithm, called *K-Landmarks*, aiming to retain clustering quality at the projected space. K-Landmarks first selects an aggregator node that picks k points (henceforth called *landmark points*) from the whole dataset of cardinality d, and projects them from R^n to R^k with FastMap [3]. The projections of the remaining $d - k$ points are computed by requesting the preservation of distances, meaning that each point projected must be at equal distance from all landmark points, both in the original and in the projection space. Our algorithm is not an adaptation of a centralized algorithm; on the contrary it is inherently distributed. Preliminary work describing the initial idea and algorithm was presented in [8]. In this paper, we present additionally the formal description of the algorithm, its geometric interpretation, the proof of convergence and new extensive experiments on various UCI datasets[1], comparing our algorithm's performance with the most promising DDR algorithms in the literature. The rest of the paper is organized as follows: in Section 2 we present a brief overview of the related work. Section 3 describes K-Landmarks, while in Section 4 the conducted experiments are presented. In Section 5 we conclude the paper and sketch future research directions.

[1] `http://www.ics.uci.edu/ mlearn/MLSummary.html`

2 Related Work

In the following presentation we mainly concentrate on distributed approaches and provide only a brief outline of the most prominent centralized algorithms. For the rest of this paper we assume that the goal is to project d data vectors defined in R^n and represented as a matrix X_{dxn} in the R^k subspace.

One of the first dimensionality reduction methods was Multidimensional Scaling (MDS) [4] that is now referenced as classic-MDS. FastMap [3] was proposed as a solution to the high computational complexity of MDS while Landmark MDS (LMDS) [2] addresses the high memory requirements of classic-MDS. The dimensionality reduction techniques widely used in practice, due to their conceptual simplicity, are Principal Components Analysis (PCA) and Singular Value Decomposition (SVD) [4]. Adaptations of the convergence criterion of MDS and PCA have resulted in the definition of Independent Component Analysis and Projection Pursuit algorithm respectively [4].

In the field of DDR we report two promising approaches, the Distributed PCA and the Distributed FastMap. The intuition of distributed PCA (DPCA) [11] is based on the aggregation of a fragmented covariance matrix, which is computed by the equation: $dC = X^T(I - d^{-1}\mathbf{1}\mathbf{1}^T)X$. For each node i possessing d_i data the following statistics are denoted: x_i as the vector of column means, k_i as the number of required principal components from node i, X_i as the local dataset and Λ_i, U_i as the matrices of the k_i largest eigenvalues and corresponding eigenvectors (in descending order) of location i. Furthermore the expression $I - d^{-1}\mathbf{1}\mathbf{1}^T$, is transformed into $(I - V) + (V - d^{-1}\mathbf{1}\mathbf{1}^T)$, where $x = d^{-1}\mathbf{1}^T X$ is the n column means vector, $\mathbf{1}$ is a vector containing ones (1s) and V is a diagonal matrix ($v_{ii} = d_i^{-1}\mathbf{1}\mathbf{1}^T$). DPCA is based on the following decomposition scheme of the covariance matrix (see [11] for details):

$$dC = \sum_{i=1}^{s} U_i \Lambda_i^2 U_i^T + \sum_{i=1}^{s} d_i (x_i - x)(x_i - x)^T \quad (1)$$

Initially, an aggregator node is selected that performs the merging and local k_i values are set. Then the statistics (d_i,k_i,x_i,Λ_i,U_i) are calculated on each network node and communicated to the aggregator. The latter, based on equation (1), calculates the global covariance matrix dC, and transmits its first k eigenvectors together with the global mean value x back to the s network nodes. Finally, each node computes its dataset embedding as follows: $D_i = (X_i - \mathbf{1}x^T)U_k$. Another approach, called Collective PCA, is considered in [6]. However it solves the problem of vertical data partitioning, while we focus on the horizontal case.

In [9] two distributed adaptations of FastMap are proposed, the One-Time Distributed FastMap and the Iterative Distributed FastMap. The former iterates on the data of each node independently, and communicates the generated pivot points to a randomly selected aggregator. Received pivots are used as input to FastMap which generates a global pivot set that is broadcasted and used for the subsequent projection of local datasets. On the other hand, the Iterative Distributed FastMap employs an iteration-by-iteration pivots computation scheme where global pivots are computed on iteration basis according to

Table 1. Assessment of the various algorithms. Number of points (d), initial dimensionality (n), projection space (k), nodes (s), and number of sampled points (f).

	Algorithmic Complexity	Memory Requirements	Addition of a new point	Network load
PCA	$O(n^2d + n^3)$	$O(n^2 + nd)$	$O(kn)$	—
DPCA	$O(n^2d_i + n^3)$	$O(n^2 + nd_i)$	$O(kn)$	$O(nsk)$
FastMap	$O(dk)$	$O((k+n)d + d^2)$	$O(k)$	—
One-Time D.FastMap	$O(d_ik)$ or $O(d_ik + sk^2)$	$O((k+n)d_i + d_i^2)$	$O(k)$	$O(skn + k^2)$
Iterative D.FastMap	$O(d_ik)$ or $O(d_ik + sk^2)$	$O((k+n)d_i + d_i^2)$	$O(k)$	$O(skn + k^2)$
LMDS	$O(kfd + f^2 + f^3)$*or* $O(kfd + f^2 + f^3 + k^2d + k^3)$	$O(f(n+k) + f^2)$*or* $O(n^2 + f^2)$	$O(kf)$	—
Distributed LMDS	$O(kfd_i + f^2)$ or $O(kfd_i + f^2 + f^3)$	$O(f(n+k))$ or $O(f(n+k) + f^2)$	$O(kf)$	$O(fn + fk)$
PAA	$O(d)$	$O(n)$	$O(1)$	0

the find-distant-objects heuristic. Although the two adaptations do not guarantee that the set of pivots selected will be identical with the ones of centralized FastMap, in large data collections they approximate well the original set and provide quality results marginally equal to the original approach.

Piecewise Aggregate Approximation (PAA) [7] is a simple and effective algorithm that can be considered as DDR, which substitutes a set of $\lfloor n/k \rfloor$ variables with their mean value. The only drawback that PAA exhibits is its dependence on the size of the rolling window ($\lfloor n/k \rfloor$). If the latter is big ($k << n$) then sharp changes in data will be lost.

The final algorithm outlined is our proposition concerning the distributed adaptation of LMDS, which we refer to as D-LMDS. LMDS is an algorithm that by construction has been developed to work with only a fraction of the total data. In our variation, the dataset is assumed to be distributed among s network nodes. We initially select an aggregator node that will be assigned the classic MDS computation. Afterwards, each node selects f_i points ($\sum_{i=1}^{s} f_i = f$) from its local dataset and forwards them to the aggregator. The latter performs classic MDS and produces their embedding in the R^k subspace. Then, the aggregator forwards all landmark points and their embeddings to network nodes. Finally, each node applies for each of the local points distance-based triangulation.

Table 1 provides a short comparative assessment of the algorithms presented above. Under certain assumptions all presented algorithms provide potential solutions to the DDR problem. DPCA or D-LMDS will deteriorate quickly and need to recompute the decomposition in the case that many new points are added. In addition, the sampling procedure in the case of D-LMDS will not depict the current state of the network and will have to be recomputed.

Similar disadvantages occur in all other algorithms, except PAA. This is because all are adaptations of already existing centralized approaches that have not been designed for distributed environments. For example, the application of the

Iterative Distributed FastMap can only take place in a context where communication between nodes as well as their availability is guaranteed (to ensure the large scale message exchange that is necessary) otherwise the synchronization of network nodes is practically impossible. Moreover, the addition of new points in an existing projection imposes the re-execution of FastMap, as its credibility lays in the pivots selection. On the other hand, PAA seems to be a viable solution.

It is therefore obvious that a new approach is required, that will combine the salient features of the aforementioned algorithms in terms of network load, algorithmic complexity and quality of results, while being immune to subsequent (after execution) changes in the processed data (i.e. massive addition or deletion of points). Moreover, the algorithm sought has to be inherently distributed and adaptable to potential network failures and topology changes. Finally it has to apply to the full extend of distributed applications, starting from controllable laboratory environment and reaching large scale P2P networks.

3 The K-Landmarks Algorithm

The K-Landmarks algorithm is a novel algorithm for DDR, designed specifically for distributed environments. It conforms to the four requirements stated in the introduction and in addition it is immune to data changes. K-Landmarks capitalizes on the general principles of D-LMDS, while differentiating in the way each step is performed and exhibiting lower complexity and network traffic. Given d resources represented as points in R^n, distributed arbitrarily in a network of s nodes, with each node storing d_i resources, we want to find a projection of the data in R^k, while retaining distances among points and the ability to achieve clustering quality comparable to the one in the original space.

Theorem 1. *A set of k points defined in R^n can be embedded in the R^k subspace without loss of distance information (zero stress projection[2]).*

Proof. The set of processed points define matrix X_{kxn}. The latter can be projected in R^k through the transformation $X'_{kxk} = X_{kxn}Q^T_{kxn}$ where the Q rows are the singular vectors of X. The relationship between the inner products matrix of the projected data and the inner products matrix of the original data is given by the following computations:

$$C' = X'_{kxk}{X'}^T_{kxk} = X_{kxn}Q^T_{kxn}(X_{kxn}Q^T_{kxn})^T = X_{kxn}X^T_{kxn} = C$$

Moreover, each cell (i,j) of C is populated by the value $x_i x_j^T$ and based on equality $C = C'$ we conclude that $x_i x_j^T = x'_i {x'_j}^T$. Then the new distance between points x'_i,x'_j is:

[2] Stress measure, defined as $\sum(d_{ij} - d_{ij})^2/d^2_{ij}$, where d_{ij} is the distance of points i,j in the original space and d_{ij} their distance in the projection space, signifies the quality of the projection in terms of distances preservation.

Algorithm 1. K-Landmarks algorithm

1: **Input:** Projection dimensionality (k), number of landmark points from node i (k_i), local dataset defined in R^n
2: **Output:** local dataset defined in R^k
3:
4: **if** node is aggregator **then**
5: Select k_i points from local dataset
6: Create landmark set, LS=∅
7: Create projected landmark set, PLS=∅
8: **for** $i = 1$ **to** s **do**
9: Receive k_i landmarks from node i
10: LS=$LS \cup k_i$
11: **end for**
12: PLS = FastMap(k, LS)
13: Communicate PLS, LS to all nodes
14: **else**
15: Select k_i points from local dataset
16: Send points to aggregator
17: Receive LS, PLS
18: **end if**
19: **for** $i = 1$ **to** all local points x **do**
20: Solve $\|x_i^{(k)} - PLS_i^{(k)}\| = \|x_i^{(n)} - LS_i^{(n)}\|$ for i=1 to k
21: **end for**
22: **return** $x_i^{(k)}$ //the projection of $x_i^{(n)}$ in R^k

$$d(x'_i, x'_j) = \sqrt{\sum_{p=1}^{k}(x'_{ip} - x'_{jp})^2} = \sqrt{\sum_{p=1}^{k}({x'}^2_{ip} + {x'}^2_{jp} - 2x'_{ip}x'_{jp})}$$
$$= \sqrt{x'_i {x'}_i^T + x'_j {x'}_j^T - 2x'_i {x'}_j^T} = \sqrt{x_i x_i^T + x_j x_j^T - 2x_i x_j^T} = d(x_i, x_j)$$

Hence, the projection of k points from R^n to R^k can be defined without loss of distance information and consequently zero stress.

Algorithm 1 is the formal description of K-Landmarks. In the first step, an aggregator node is selected. The latter uses k landmark points (LS) sampled from the network (line 9) and projects them to R^k with FastMap (line 12). The original set of landmark points and the generated mapping (PLS) are forwarded to all nodes (line 13), which in turn project local points independently (line 20). The successful projection of a point in R^k maintains its distances from the landmark points both in the original and in the projection space.

The equation: $\|x_i^{(k)} - PLS_i^{(k)}\| = \|x_i^{(n)} - LS_i^{(n)}\|$ represents a hypersphere centered at $PLS_i^{(k)}$ with radius $||x_i^{(n)} - LS_i^{(n)}||$. The algorithm searches the common trace of all k hyperspheres, which is the projection of point x_i in the embedding space. The result is obtained by solving the above system of non-linear equations with the Newton method. An example of the algorithm is depicted in Fig. 1. Note that the hyperspheres defined by the algorithm expose two common traces, which are symmetric to the line defined by points $A'B'$. This is due to the

fact that the coordinates of each point result from a square root computation. We choose to retain only positive results and therefore the implementation of our algorithm discards the second solution. In any case, one can select either the projection laying on plane $\Pi 1$ or $\Pi 2$ without affecting the final result, as long as the same plane selection algorithm is employed for point D.

Theorem 2. *For any non linear system of equations defined by K-Landmarks, the Newton method produces a solution if the triangular inequality is sustained in the original space.*

Proof. For any point C of the initial space and any pair of landmark points A and B, a triangle ABC is defined. Without loss of generality, we assume that $\|\overrightarrow{CB}\| \leq \|\overrightarrow{CA}\|$ and based on the triangular inequality we derive:

$$\|\overrightarrow{CA}\| - \|\overrightarrow{CB}\| \leq \|\overrightarrow{AB}\| \leq \|\overrightarrow{CA}\| + \|\overrightarrow{CB}\| \tag{1}$$

The system defined for the projection is the following:

$$\|\overrightarrow{CA}\| = \|\overrightarrow{C'A'}\| \text{ and } \|\overrightarrow{CB}\| = \|\overrightarrow{C'B'}\|$$

This system has no solution if there exists no common trace between the aforementioned hyperspheres. This is translated to: $\|\overrightarrow{A'B'}\| < \|\overrightarrow{C'A'}\| - \|\overrightarrow{C'B'}\|$ or $\|\overrightarrow{A'B'}\| > \|\overrightarrow{C'A'}\| + \|\overrightarrow{C'B'}\|$ and equally:

$$\|\overrightarrow{A'B'}\| > \|\overrightarrow{CA}\| + \|\overrightarrow{CB}\| \text{ (2) or } \|\overrightarrow{A'B'}\| < \|\overrightarrow{CA}\| - \|\overrightarrow{CB}\| \tag{3}$$

However, based on [10] and Theorem 1 we obtain a zero stress projection from FastMap, thus deriving:

$$\|\overrightarrow{A'B'}\| = \|\overrightarrow{AB}\| \tag{4}$$

Consequently, based on (1), (4) we conclude that equations (2), (3) are never true, meaning that the system in question always has a solution (there always

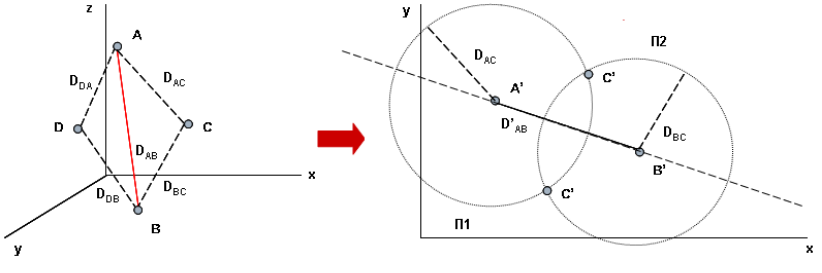


Fig. 1. Geometrical interpretation of the K-Landmarks algorithm. Projection of point C from R^3 (left) to R^2 (right). A, B are the landmark points.

Table 2. K-Landmarks evaluation matrix

	Algorithmic Complexity	Memory Requirements	Addition of a new point	Network load
K-Landmarks	$((d_i - k_i)k^3/3)$	$O(kn + k^2)$	$O(k^3/3)$	$O(nk + k^2)$

exists a projection) provided that the triangular inequality is sustained in the original space.

One thing that has not yet been discussed is the selection of initial points. K-Landmarks employs three different initialization techniques namely: random, MaxMin and MaxDist. In random selection each node selects k_i points from its dataset randomly. The MaxMin heuristic [2] enforces the selection of points that maximize the minimum distance from any of the already selected landmark points while the MaxDist selects the furthest from the existing landmark points. Both heuristics select their first point randomly and are applied on local datasets for the retrieval of the k_i points requested by the algorithm.

To sum up, the proposed algorithm differs significantly from other widely employed DDR approaches since it achieves the projection of the vast majority of points independently from the rest, implying that only the (few) landmark points' projection will be done in a centralized manner. Moreover, the projection remains unaffected by subsequent additions of data points. This is due to the fact that any new point will be mapped analogously close or far from the landmark points depending on its distance from the latter in the original space. Consequently, no re-computation of the projection is needed in order to guarantee projection quality preservation. Furthermore, the minimization criterion employed by the algorithm ($\sum_{|LS|} |distance_{orig} - distance_{new}|$) is applied to each point independently, contrary to the widely employed stress that is applied to the whole dataset. Finally, the network load imposed (see Table 2) is lower than the load of other algorithms.

Apart from the above, the proposed algorithm is inherently distributed, in contrast to the other distributed algorithms described in Section 2. One can imagine its usage in a P2P environment. These networks exhibit certain intrinsic peculiarities, such as instability, bandwidth restrictions, etc. If K-Landmarks is used, the sampling procedure will be carried out once in the lifetime of the network and the result will be forwarded to all nodes entering the network at any time. The added value of the approach is apparent, as its immunity to additions saves both local and network resources.

4 Experiments

We study the comparative performance of K-Landmarks on various datasets (Table 3) from the UCI Machine Learning Repository. We highlight the use of datasets both of higher dimensionality (up to 617 dimensions) and cardinality

(up to 1559 objects) compared to relevant work [3,9]. Our goal is to achieve results of quality close to well-known centralized and distributed algorithms [3].

4.1 Experimental Setup

The experimental scenario involves clustering various data sets before and after the application of a DDR algorithm. The quality metrics we use are the stress value and the clustering quality maintenance. Stress evaluates the quality of the projection in terms of distances' preservation, while the second measure enables us to observe how DDR affects the clustering quality. We capitalize on the well known clustering quality index F-Measure [1] and define clustering quality preservation as the ratio of the F-measure values before and after the DDR process, i.e. $\frac{F-Measure(R^k)}{F-Measure(R^n)}$. The clustering algorithm employed is K-Means [1].

The experiments also allow measuring the effect of changing the projection dimensionality on the stress and F-measure values. Each dataset was evaluated with 5 different projection dimensions (5 points for each algorithm on the charts), which were defined as a fraction of the initial dimensionality of the dataset. Each time the projection dimensionality is increased by 2% of the initial dimensions.

Assume a network of s nodes, with d data vectors distributed evenly among them. The vectors are defined in R^n and projected in R^k. The algorithms we use in comparison to K-Landmarks are PCA, FastMap, Distributed FastMap, Distributed PCA, Distributed LMDS and PAA. The notation employed in the diagrams is the following: KL refers to K-Landmarks, LMDS to the D-LMDS and DFM to distributed FastMap. DPCA uses the k principal eigenvectors generated by each peer separately, while K-Landmarks and LMDS randomly select from each node $\lceil k/s \rceil$ and $\lceil k+1/s \rceil$ points respectively. The reason for the selection of $\lceil k+1/s \rceil$ points for LMDS lays in the original publication [2], where it is advised to choose at least $k+1$ landmarks. Also, the Newton method employed by K-Landmarks is initialized with the k first coordinates of each processed vector in the original space. All experiments have been carried out on a commodity 2.4GHz Pentium IV machine with 768MB of RAM. The results are mean values of 50 executions and each setup simulates a network of $s = 20$ nodes.

4.2 Results

In the first set of experiments, we measure for each algorithm and target dimensionality value the F-measure quality preservation (y-axis) versus the respective stress value of the projection (x-axis). Our aim is to identify the DDR algorithms that exhibit low stress while maintaining clustering quality.

In Fig. 2, experiments on the "Ionosphere" dataset show that K-Landmarks outperforms all its competitors, even the centralized approaches such as PCA and FastMap. Most of the K-Landmarks measurements reflect extremely low stress and high clustering quality maintenance, close to or higher than 100% (i.e.

[3] An extended set of experiments is available at `http://www.db-net.aueb.gr/cdoulk/content/papers/xKLandmarks.pdf`

Table 3. Datasets used in the experiments

Dataset	Objects	Dimensions	Classes	Description
Ionosphere	351	34	2	Radar observations
Isolet5	1559	617	26	Letters of the alphabet
P.I.Diabetes	768	8	2	Medical observations
Segmentation	2000	19	7	Outdoor images segments
Synthetic control	600	60	6	Randomly generated data

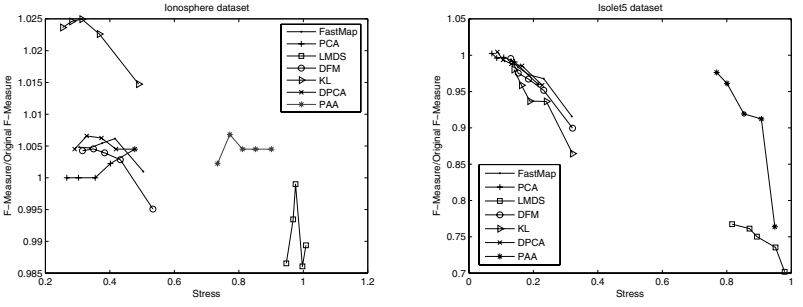


Fig. 2. Clustering Quality vs. stress for the "Ionosphere" and "Isolet" datasets

centralized clustering quality). This is due to the empty space phenomenon and the curse of dimensionality. LMDS exhibits the worst behavior proving rather unstable. Experiments on "Isolet5" provide better insight. All algorithms except from PAA and LMDS performed similarly, with PCA performing better. PAA also exhibits satisfactory results in clustering quality maintenance, but higher stress than the rest. Finally, LMDS is rather unsatisfactory in both quality axes.

The experiments on "pima indians diabetes" and "synthetic control" (Fig. 3) gave similar results to the "ionosphere" dataset. K-Landmarks achieves the best overall performance, showing both low stress and high clustering quality maintenance. PAA and LMDS achieve marginally equal clustering quality results, compared to the rest of the approaches, but with higher stress values. However PAA proves to be better than LMDS, when both measures are considered.

In Fig. 4, we measure F-measure maintenance and stress for different projection dimensionality values (k) for the "image segmentation" dataset. The aim is to study the effect of k on these two measures. Stress values decrease monotonously as k increases (Fig. 4 left), because the ability to express distances between data increases too. The same general tendency appears in the case of clustering quality maintenance (Fig. 4 right). For the majority of the algorithms, clustering quality maintenance ameliorates with k. The conclusion is that DDR algorithms perform better with increasing projection dimensionality. We clarify that the average variance of all measured values in the 50 executions for all datasets is in the order of 10^{-4}, showing the stability of our algorithm.

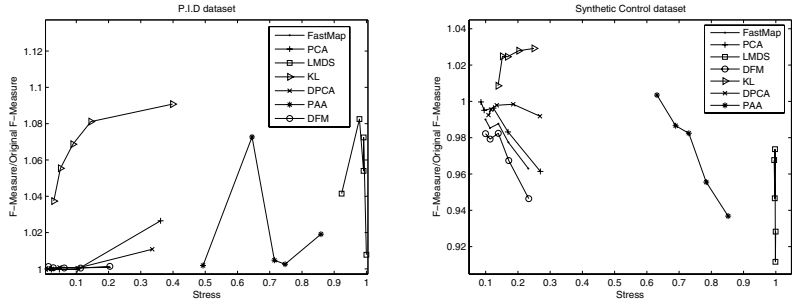


Fig. 3. Results from experiments on "P.I.diabetes" and "synthetic control" dataset

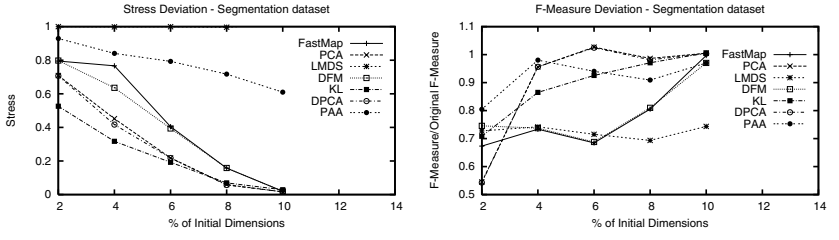


Fig. 4. Stress and F-Measure deviation in the "image segmentation" dataset

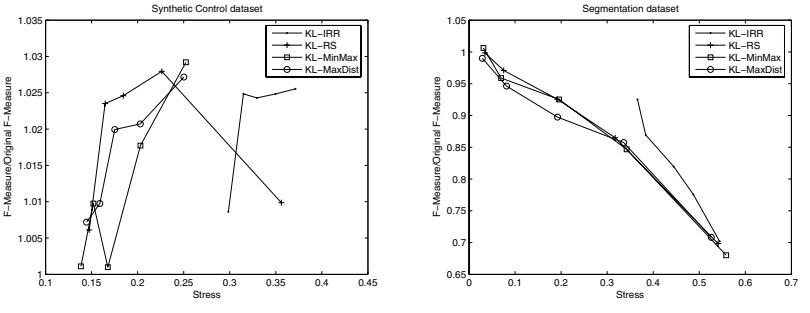


Fig. 5. K-Landmarks immunity to the addition of data

In the last set of experiments, we demonstrate the robustness of K-Landmarks to retain clustering quality, with regards to new data points added to the dataset. In Fig. 5, we study the algorithm's different initialization setups. The new setup depicted is KL-IRR, in which the algorithm was initiated with k points that were randomly generated and did not belong to the dataset (we remind here the 3 other initialization setups: random, MaxMin and MaxDist).

The datasets are subsequently added to the projection. In the "synthetic control" dataset, we added 600 points, while for the "image segmentation" dataset we added 2000. In both cases points were inserted after the algorithm's execution with the initial k landmarks. The obtained results exhibit an aggravation tendency in the stress measure, yet the clustering quality remains almost the same. These experiments show K-Landmarks is barely affected by the massive insertion of data points and can guarantee clustering quality maintenance with a slight loss in the preservation of the original distances. Another conclusion, clearly depicted on the right diagram, is that the results obtained by our algorithm are not affected by the way initial points are selected. Therefore, based on our experiments, we suggest the random selection initialization scheme (KL-RS).

5 Conclusions and Future Work

In this paper we proposed a new effective, inherently distributed, algorithm for DDR, K-Landmarks. We compared experimentally our approach to well known DDR approaches with regards to stress and clustering quality preservation. The results show that K-Landmarks is a robust algorithm outperforming existing DDR approaches. Moreover, it is comparable and sometimes even superior to centralized methods. Clustering quality is retained with dimensionality reduction, in spite of the small loss in distance preservation. Our future work will mainly focus on the evaluation of K-Landmarks with text data and its combination with distributed clustering approaches.

References

1. S. Chakrabarti. *Mining the Web - Discovering Knowledge from Hypertext Data*. Morgan Kaufmann Publishers, 2003.
2. V. de Silva and J. B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford Mathematics, 2004.
3. C. Faloutsos and D. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of SIGMOD'95*, 1995.
4. I. K. Fodor. A survey of dimension reduction techniques. Technical report, LLNL, UCRL-ID-148494, 2002.
5. A. Hinneburg, C. Aggarwal, and D. Keim. What is nearest neighbor in high dimensional spaces? In *Proceedings of VLDB'00*, 2000.
6. H. Kargupta, W. Huang, K. Sivakumar, and E. L. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.
7. E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

8. P. Magdalinos, C. Doulkeridis, and M. Vazirgiannis. A novel effective distributed dimensionality reduction algorithm. In *Proceedings of FSDM'06*, 2006.
9. F. N.Abu-Khzam, N. Samatova, G. Ostrouchov, M. A.Langston, and A. Geist. Distributed dimension reduction algorithms for widely dispersed data. In *Int. Conference on Parallel and Distributed Computing Systems (PDCS'02)*, 2002.
10. J. C. Platt. FastMap, MetricMap and Landmark MDS are all Nyström algorithms. In *10th International Workshop on Artificial Intelligence and Statistics*, 2005.
11. Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist. Principal component analysis for dimension reduction in massive distributed data sets. In *Proceedings of the 5th International Workshop on High Performance Data Mining*, 2002.

The Discrete Basis Problem

Pauli Miettinen[1], Taneli Mielikäinen[1], Aristides Gionis[1], Gautam Das[2,⋆], and Heikki Mannila[1]

[1] HIIT Basic Research Unit, Department of Computer Science, University of Helsinki, P.O. Box 68, FIN-00014, Finland
{pamietti, tmielika, gionis, mannila}@cs.Helsinki.FI
[2] Computer Science and Engineering Department, University of Texas at Arlington, Arlington TX 76019, USA
gdas@cse.uta.edu

Abstract. Matrix decomposition methods represent a data matrix as a product of two smaller matrices: one containing basis vectors that represent meaningful concepts in the data, and another describing how the observed data can be expressed as combinations of the basis vectors. Decomposition methods have been studied extensively, but many methods return real-valued matrices. If the original data is binary, the interpretation of the basis vectors is hard. We describe a matrix decomposition formulation, the Discrete Basis Problem. The problem seeks for a Boolean decomposition of a binary matrix, thus allowing the user to easily interpret the basis vectors. We show that the problem is computationally difficult and give a simple greedy algorithm for solving it. We present experimental results for the algorithm. The method gives intuitively appealing basis vectors. On the other hand, the continuous decomposition methods often give better reconstruction accuracies. We discuss the reasons for this behavior.

1 Introduction

Given an $n \times m$ matrix $\boldsymbol{C}$ and an integer $k < m$, classical matrix decomposition methods aim at finding an $n \times k$ matrix $\boldsymbol{S}$ and a $k \times m$ matrix $\boldsymbol{B}$ such that $\boldsymbol{C}$ can be approximately represented as the product of $\boldsymbol{S}$ and $\boldsymbol{B}$. The decomposition method represents the data by using k components: the matrix $\boldsymbol{B}$ tells how the components are related to the original attributes (columns), and the matrix $\boldsymbol{S}$ indicates how strongly each component is related to each row.

Singular value decomposition (SVD) [1] and nonnegative matrix factorization (NMF) [2] are typical examples of decomposition methods; the difference between the two is that NMF assumes that $\boldsymbol{C}$ is nonnegative and requires that $\boldsymbol{S}$ and $\boldsymbol{B}$ are nonnegative. Other matrix decomposition methods include latent Dirichlet allocation (LDA) [3] and multinomial PCA [4]; see Section 3 for additional discussion of related work.

⋆ Part of the work was done while the author visited HIIT Basic Research Unit, Department of Computer Science, University of Helsinki, Finland.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 335–346, 2006.
© Springer-Verlag Berlin Heidelberg 2006

These (and other) matrix decomposition methods allow the matrices $\boldsymbol{S}$ and $\boldsymbol{B}$ to contain arbitrary real numbers. However, if the input matrix $\boldsymbol{C}$ is binary, it is natural to require that $\boldsymbol{S}$ and $\boldsymbol{B}$ are also binary. In this paper we consider the matrix decomposition problem created by this requirement. In this case the combination operation of matrices $\boldsymbol{S}$ and $\boldsymbol{B}$ is the Boolean matrix product (i.e., the matrix product in the semiring of Boolean $\wedge$ and $\vee$).

The intuition behind considering Boolean operations is as follows. Consider a course enrollment dataset in a CS department. Such a dataset indicates which students enroll to which courses. Naturally, courses are divided into specialization areas. A student X interested in the `Systems` specialization needs to take, among others, courses on {`Operating Systems, Programming languages`}, and a student Y interested in the `Software` specialization needs to take courses on {`Compilers, Programming languages`}. On the other hand, a student Z interested in combining both of the above two specializations should take all courses {`Compilers, Operating systems, Programming languages`} (among others). The point is that student Z should (obviously) take `Programming languages` only once. Thus the set union operation is more appropriate for describing the actual data from the basis vectors (specialization areas).

Following the intuition in the previous example, we formulate the problem of finding a decomposition into binary matrices that give the best approximation to the input matrix. We call this problem the *Discrete Basis Problem* (DBP). We show that DBP is NP-hard and it cannot be approximated unless P = NP.

We give a simple greedy algorithm for solving the DBP and assess its empirical performance. We show that the algorithm produces intuitively appealing basis vectors. On the other hand, the continuous decomposition methods often give better reconstruction accuracies and are also stronger in providing predictive features for classification. We discuss the reasons for this behavior.

The rest of the paper is organized as follows. In Section 2 we formally define the Discrete Basis Problem and in Section 3 we review related work. In Section 4 we compare continuous and discrete matrix decomposition approaches. In Section 5 we discuss issues related to the computational complexity of DBP. In Section 6 we present our greedy algorithm, and in Section 7 we report experimental results. Finally, Section 8 is a short conclusion.

2 Problem Definition

Consider an $n \times m$ binary matrix $\boldsymbol{C}$. The rows of the matrix represent observations and the columns the attributes of the dataset. For instance, in a document corpus dataset, rows are documents and columns are words, and $\boldsymbol{C}_{ij} = 1$ denotes that the i'th document contains the j'th word.

A *basis vector*, intuitively, represents a set of correlated attributes. In the document corpus example, a basis vector corresponds to a set of words that constitute a *topic*. The DBP formulation aims at discovering the topics that are present in the dataset, and also discovering how each observation (document) in the dataset can be expressed by a combination of those topics.

Let $\boldsymbol{S}$ and $\boldsymbol{B}$ be binary matrices of dimensions $n \times k$ and $k \times m$, respectively. Let $\boldsymbol{P} = \boldsymbol{S} \circ \boldsymbol{B}$ denote the ($n \times m$ matrix) Boolean product of $\boldsymbol{S}$ and $\boldsymbol{B}$, i.e., the matrix product with addition defined by $1 + 1 = 1$. The i'th row of $\boldsymbol{P}$ is the logical OR of the rows of $\boldsymbol{B}$ for which the corresponding entry in the i'th row of $\boldsymbol{S}$ is 1. Intuitively, $\boldsymbol{S}$ is the *usage matrix*, i.e., it contains information about which topics appear in each observation, and $\boldsymbol{B}$ is the *basis vector matrix*, i.e., it contains information about which attributes appear in each topic.

The DBP seeks k binary basis vectors such that the binary data vectors can be represented by using disjunctions of the basis vectors. The key aspect of the formulation is that both decomposition matrices, $\boldsymbol{S}$ and $\boldsymbol{B}$, are required to be binary, and are thus more easily interpretable than arbitrary real matrices. Formally, the DBP is defined as follows.

Problem 1 (The Discrete Basis Problem). Given a binary $n \times m$ matrix $\boldsymbol{C}$ and a positive integer $k < \min\{m, n\}$, find an $n \times k$ binary matrix $\boldsymbol{S}$ and a $k \times m$ binary matrix $\boldsymbol{B}$ that minimize

$$|\boldsymbol{C} - \boldsymbol{S} \circ \boldsymbol{B}| = \sum_{i=1}^{n} \sum_{j=1}^{m} |c_{ij} - (\boldsymbol{S} \circ \boldsymbol{B})_{ij}| \,. \tag{1}$$

3 Related Work

Probably the best-known method to decompose a matrix is the Singular Value Decomposition (SVD) [1]. The SVD decomposes a matrix $\boldsymbol{A}$ into the form $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal matrices and $\boldsymbol{\Sigma}$ is a diagonal matrix with positive entries—the singular values of $\boldsymbol{A}$. SVD gives the *optimal* rank-k approximation of the matrix $\boldsymbol{A}$ (simply by setting all but the k largest singular values to 0). Optimality of SVD means that the approximation produced by SVD is the best with respect to the squared reconstruction error and using normal matrix multiplication for real matrices. SVD has been widely used in data mining for matrix simplification and topic identification.

One problem with SVD is that the factor matrices can also contain negative values that are difficult to interpret (see also Section 4). To overcome this problem, and also to avoid the restriction to orthogonal matrices, Lee and Seung proposed a method known as non-negative matrix factorization (NMF) [2]. While NMF does not minimize the global squared reconstruction error, the existing algorithms for NMF converge to the local optima [5].

In addition to SVD and NMF, many other matrix decomposition methods have been proposed, most of which are based on probabilistic models. Such methods include multinomial PCA [4], probabilistic Latent Semantic Indexing [6], Latent Dirichlet Allocation [3], aspect Bernoulli models [7], and topic models [8]. The last two models are closest to DBP as they are designed for binary data.

Also hierarchical descriptions of binary data have been studied: the Proximus framework constructs a hierarchical clustering of rows of a given binary matrix [9] and hierarchical tiles are probabilistic models hierarchically decomposing a binary matrix into almost monochromatic 0/1 submatrices [10].

Tiling transaction databases (i.e., binary matrices) is another line of related research [11]. A tiling covers a given binary matrix with a small number of submatrices full of 1's. The main difference to DBP is that no 0's can be covered in a feasible tiling. Methods have been developed for finding also large approximate tiles, for example fault-tolerant patterns [12] and conjunctive clusters [13], but obtaining an accurate description of the whole dataset with a small number of approximate tiles has not been studied previously explicitly.

Boolean factorization, i.e., factoring Boolean functions [14], is an important part of logic synthesis. Rectangular coverings of Boolean matrices are one method used to obtain good factorizations. However, the weight functions used and the acceptance of noise are different to those of our work.

Finally, in co-clustering (or bi-clustering) the goal is to cluster simultaneously both dimensions of a matrix [15]. A co-cluster is thus a tuple (R, C), R giving the indices of rows and C giving the indices of columns. Decomposing a Boolean matrix into two matrices can be seen as a co-clustering of binary data where the clusters can overlap. Different methods for co-clustering have been proposed, see, for example, work of Banerjee, Dhillon and others [15].

4 Continuous and Discrete Decompositions

In this section we discuss the properties of continuous and discrete approaches to matrix decomposition, and in particular the properties of SVD as compared to those of DBP.

In SVD the resulting matrices, $\boldsymbol{U}$ and $\boldsymbol{V}$, have real-valued and even negative entries, so they do not necessarily have an intuitive interpretation. As an example, consider the case of matrix $\boldsymbol{C}$ and its rank-2 SVD decomposition:

$$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad \boldsymbol{U} = \begin{pmatrix} 0.50 & 0.71 \\ 0.71 & 0 \\ 0.50 & -0.71 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} 2.41 & 0 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{V} = \begin{pmatrix} 0.50 & 0.71 \\ 0.71 & 0 \\ 0.50 & -0.71 \end{pmatrix}.$$

The basis vectors in $\boldsymbol{V}$ are not the easiest to interpret. Matrix $\boldsymbol{C}$ has rank 3, and the approximation to $\boldsymbol{C}$ produced by SVD with rank-2 decomposition is

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{pmatrix} 1.10 & 0.85 & 0.10 \\ 0.85 & 1.21 & 0.85 \\ 0.10 & 0.85 & 1.10 \end{pmatrix}.$$

By the optimality of SVD, this is the best that can be achieved by looking at real matrices and squared error. On the other hand, DBP produces the representation

$$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$

which has no error and is easy to understand.

As noted above, SVD produces optimal rank-k representations of matrices with respect to the Frobenius norm (sum of squares of elements). It is also relatively fast to compute, requiring time $O(nm \min\{n, m\})$ [1].

Optimality for arbitrary matrices is not the whole story, however. For binary matrices, one can study two types of ranks. The *real rank* $r_R(\boldsymbol{C})$ of a binary matrix $\boldsymbol{C}$ is simply the smallest value of k such that $\boldsymbol{C} = \boldsymbol{SB}$ with an $n \times k$ matrix $\boldsymbol{S}$, a $k \times m$ matrix $\boldsymbol{B}$, and using normal matrix multiplication. The *Boolean rank* $r_B(\boldsymbol{C})$ of $\boldsymbol{C}$ is the smallest k such that $\boldsymbol{C} = \boldsymbol{S} \circ \boldsymbol{B}$, where $\boldsymbol{S}$ is an $n \times k$ matrix, $\boldsymbol{B}$ is a $k \times m$ matrix, and the matrix multiplication is Boolean. It can be shown that there are matrices $\boldsymbol{C}$ for which $r_R(\boldsymbol{C}) < r_B(\boldsymbol{C})$ and vice versa [16]. The complement of the identity matrix of size $n \times n$ is an example where $r_B(\boldsymbol{C}) = O(\log n)$, but $r_R(\boldsymbol{C}) = n$ [16]. This shows that while SVD can use the space of reals, DBP can take advantage of the properties of Boolean operations to achieve much smaller rank than SVD. Empirical experiments on generated data support this conclusion. Thus it is not a priori obvious that SVD will produce more concise representations than the Boolean methods.

The concepts of real and Boolean rank discuss the exact representation of the matrix $\boldsymbol{C}$, and we are more interested in the approximate representation. One could define the ε-ranks $r_R^\varepsilon(\boldsymbol{C})$ and $r_B^\varepsilon(\boldsymbol{C})$ as the smallest k such that there is a representation of $\boldsymbol{C}$ as $\boldsymbol{SB}$ or $\boldsymbol{S} \circ \boldsymbol{B}$ with $\boldsymbol{B}$ being a $k \times m$ matrix and $|\boldsymbol{C} - \boldsymbol{SB}| < \varepsilon$. Even less seems to be known about such concepts than about exact real and Boolean ranks. One goal of our paper is to investigate empirically and theoretically whether the Boolean decompositions are feasible alternatives of the continuous methods.

5 Computational Complexity of DBP

The DBP is an optimization problem: find the matrix decomposition into k basis vectors that minimizes the representation error according to the definition of Problem 1. To put the problem in the perspective of complexity theory, we formulate the decision version of the problem. This is defined as in Problem 1, but additionally we are given a target cost t and the task is to decide whether there is a decomposition of the input binary matrix $\boldsymbol{C}$ into binary matrices $\boldsymbol{S}$ and $\boldsymbol{B}$ that yields an error at most equal to t.

The problem is NP-hard as the Set Basis Problem (SBP) [17, problem SP7] is a special case of the decision version of the DBP.

Problem 2 (The Set Basis Problem). Given a collection $\mathcal{C}$ of subsets of a finite universe U and a positive integer k, decide whether or not there is a collection $\mathcal{B} \subseteq 2^U$ of at most k sets ($|\mathcal{B}| \leq k$) such that for every set $C \in \mathcal{C}$ there is a subcollection $\mathcal{B}_C \subseteq \mathcal{B}$ with $\bigcup_{B \in \mathcal{B}_C} B = C$.

More specifically, for any instance of SBP there is an equivalent instance of DBP with $t = 0$, even when only the matrix $\boldsymbol{B}$ is requested. The NP-hardness can also be shown by observing that the Biclique Covering Problem is a special case of DBP with $t = 0$ where both $\boldsymbol{S}$ and $\boldsymbol{B}$ are needed. It is immediate that DBP is in NP. Thus we have:

Theorem 1. *The decision version of* DBP *is* NP*-complete.*

The reduction from SBP to DBP with $t = 0$ implies also the following simple inapproximability result:

Theorem 2. *DBP cannot be approximated within any factor in polynomial time, unless* P = NP*.*

The problem of solving the whole decomposition of the matrix for given basis vectors, i.e., finding the matrix $\boldsymbol{S}$ for given $\boldsymbol{B}$ and $\boldsymbol{C}$, can be solved by a straightforward algorithm in time $O(2^k mn)$ where k is the number of basis vectors (i.e., the number of rows in $\boldsymbol{B}$): Each of the n rows in $\boldsymbol{C}$ can be decomposed independently and there are only 2^k different ways to choose a subset of basis vectors. Thus the problem of finding the optimal decomposition after the basis vector matrix is known, is in the class of *fixed-parameter tractable* problems (see [18]).

6 The Algorithm

In this section we give a greedy algorithm for DBP. The basic idea of the algorithm is to exploit the correlations between the columns. First, the associations between each two columns are computed. Second, the associations are used to form candidate basis vectors. Third, a small set of candidate basis vectors are selected in a greedy way to form the basis.

In the rest of the section we denote a row vector of a matrix $\boldsymbol{M}$ by $\boldsymbol{m}_i$, a column vector by $\boldsymbol{m}_{\cdot i}$ and a matrix entry by m_{ij}. The confidence of an association between the i-th and j-th column is defined as in association rule mining [19], i.e., $c(i \Rightarrow j) = \langle \boldsymbol{c}_{\cdot i}, \boldsymbol{c}_{\cdot j} \rangle / \langle \boldsymbol{c}_{\cdot i}, \boldsymbol{c}_{\cdot i} \rangle$, where $\langle \cdot, \cdot \rangle$ is the vector inner product operation. An association between columns i and j is τ-strong if $c(i \Rightarrow j) \geq \tau$.

We construct an *association matrix* $\boldsymbol{A}$ where row $\boldsymbol{a}_i$ consists of 1's in columns j such that $c(i \Rightarrow j) \geq \tau$. Each row of $\boldsymbol{A}$ is considered as a candidate for being a basis vector. The threshold τ controls the level of confidence required to include an attribute to the basis vector candidate, and it is assumed that τ is a parameter of the algorithm.

The DBP objective function, described by (1), penalizes equally for both types of errors: for 0 becoming 1 in the approximation, and for 1 becoming 0. We have found that in practice the results of DBP can be improved if we distinguish between these two types of error. Thus we introduce weights w^+ and w^- that are used to reward for covering 1's and penalize for covering 0's, respectively. Clearly, without loss of generality, we can assume that $w^- = 1$.

The basis vectors are selected from the matrix $\boldsymbol{A}$ and the columns of the usage matrix $\boldsymbol{S}$ are fixed in a greedy manner as follows. Initially $\boldsymbol{B} = 0^{k \times m}$ and $\boldsymbol{S} = 0^{n \times k}$. The basis $\boldsymbol{B}$ is updated in the iteration l by setting the row $\boldsymbol{b}_l$ to be the row $\boldsymbol{a}_i$ in $\boldsymbol{A}$ and the column $\boldsymbol{s}_{\cdot l}$ to be the binary vector maximizing

$$\begin{aligned} \mathit{cover}(\boldsymbol{B}, \boldsymbol{S}, \boldsymbol{C}, w^+, w^-) = {} & w^+ |\{(i,j) : c_{ij} = 1, (\boldsymbol{S} \circ \boldsymbol{B})_{ij} = 1\}| \\ & - w^- |\{(i,j) : c_{ij} = 0, (\boldsymbol{S} \circ \boldsymbol{B})_{ij} = 1\}|, \end{aligned}$$

Algorithm 1. An algorithm for the DBP using association rules

Input: Matrix $\boldsymbol{C} \in \{0,1\}^{n \times m}$ for data, positive integer $k < \min\{n, m\}$, threshold value $\tau \in]0, 1]$, and real-valued weights w^+ and w^-.
Output: Matrices $\boldsymbol{B} \in \{0,1\}^{k \times m}$ and $\boldsymbol{S} \in \{0,1\}^{n \times k}$.

```
1: function Association(C, k, τ, w+, w−)
2:     for i = 1, ..., n do                ▷ Construct the association matrix A row by row.
3:         a_i ← {j : c(i ⇒ j) ≥ τ}
4:     B ← 0^{k×m}
5:     for l = 1, ..., k do                ▷ Select the k basis vectors from A.
6:         b_l ← a_i and s_l ← {0,1}^n maximizing cover(B, S, C, w+, w−)
7:     return B and S
```

which can be considered as the "profit" of describing $\boldsymbol{C}$ using the basis $\boldsymbol{B}$ and the decomposition $\boldsymbol{S}$.

The association matrix can be constructed in time $O(nm^2)$ and a single discrete basis vector can be obtained in time $O(nm^2)$. Thus, Algorithm 1 has time complexity $O(knm^2)$. The run-time can be improved in practice by using upper bounds and approximations for the confidences.

The algorithm has two parameters that control the quality of results: the threshold τ, and weight w^+ (again assuming that $w^- = 1$). The straightforward way to set the parameters is to try several different possibilities and take the best. Alternatively the weight w^+ can be used to express different valuations for covering 1's and 0's.

Unfortunately there exist cases, where the algorithm is able to find only suboptimal solution. For example, if all 1's in some basis vector occur in some other basis vectors, then the algorithm is unable to find that basis vector.

7 Experimental Results

We have performed tests using Algorithm 1 on generated and real-world datasets. The goals of the experiments are (*i*) to verify whether DBP produces intuitive basis vectors, (*ii*) to check whether DBP can reconstruct basis vectors used to generate artificial data, and (*iii*) to compare the reconstruction accuracy of DBP against SVD and NMF both for real and generated data.

7.1 Data and Error Measures

Generated data. We generated three sets of data to test the effects of (*i*) noise, (*ii*) overlap between basis vectors, and (*iii*) input size. First, a set of basis vectors was generated; then random subsets of these basis vectors were used to generate the data rows; finally, random uniform noise was added. Details on the parameters used to generate the three sets of data are shown in Table 1.

Real data. The real data consists of the following datasets: NSF Abstracts, 20 Newsgroups, Digits, and Courses. Details of the datasets are given in Table 2.

Table 1. Details on generated datasets. Each row represents a set of generated datasets. #bv: number of basis vectors; #bv/row: average number of basis vectors used to generate each data row; 1s/bv: number of 1's per basis vector; noise: number of entries flipped in the data as a percentage of the total number of 1's.

dataset	rows	columns	#bv	#bv/row	1s/bv	noise
set 1	1000	500	12	4	50	5–40%
set 2	1000	500	12	4, 8	25–200	0%
set 3	1K–16K	1K–16K	10–160	5–80	0.5K–80K	0%

NSF Abstracts[1] contain document–word information on a collection of project abstracts submitted for funding by NSF. 20 Newsgroups[2] is a collection of approximately 20000 newsgroup documents across 20 different newsgroups [20]. Digits[3] is a collection of 1000 binary images of handwritten digits [21]. Courses is a student–course dataset of courses completed by the CS students of the University of Helsinki. A random sample of NSF Abstracts and 20 Newsgroups was used for comparisons between different algorithms due to memory constraints of SVD and NMF implementations.

Table 2. Attributes of the real-world datasets

dataset	rows	columns	1s in data	avg. 1s/row	avg. 1s/column
NSF Abstracts	12 841	4 894	564 462	43.96	115.34
20 Newsgroups	10 000	5 163	455 526	45.55	88.23
Digits	2 000	240	291 654	145.83	1 215.23
Courses	2 405	615	52 739	21.92	85.75

Error measures. We use two measures to quantify the error of the approximation: sum-of-absolute-values distance d_1 and Frobenius distance d_2, defined as

$$d_1(\boldsymbol{A}, \boldsymbol{B}) = \sum_i \sum_j |a_{ij} - b_{ij}| \quad \text{and} \quad d_2(\boldsymbol{A}, \boldsymbol{B}) = \sqrt{\sum_i \sum_j (a_{ij} - b_{ij})^2}.$$

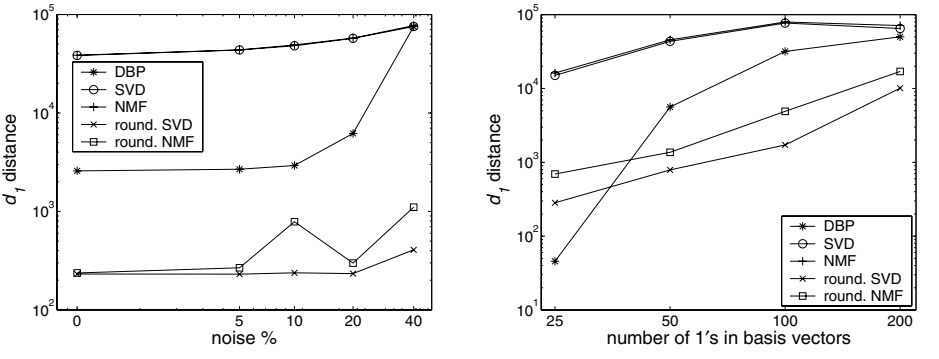
7.2 Results

Reconstructing the basis vectors from generated data. We studied the effects of noise and overlap between basis vectors to the reconstruction error. The main measure was the d_1 distance. Algorithm 1 was compared against SVD and NMF. For SVD and NMF we also used the knowledge that the matrix is supposed to be binary, and rounded the reconstructed matrix *before* computing the error with respect to the original matrix. Values smaller than 0.5 are rounded to 0 and

[1] http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html
[2] http://people.csail.mit.edu/jrennie/20Newsgroups/
[3] http://www.ics.uci.edu/ mlearn/databases/optdigits/

values greater than 0.5 are rounded to 1. We call these methods *rounded* SVD and *rounded* NMF.

The effects of noise are illustrated in Figure 1(a). Lines for plain SVD and NMF coincide at the top of the figure, partly because of the logarithmic scale. In general, plain SVD and NMF are the worst, rounded SVD and NMF are the best, and DBP is in between. Additionally, all methods seem to be rather immune to small amounts of noise. With one input set, rounded NMF converged far from global optimum, thus causing a peak in the graph.



(a) (b)

Fig. 1. Reconstruction errors using d_1 as a function of (a) noise (dataset 1), and (b) 1's in basis vector (dataset 2). Points in plots represent the mean error over five random data matrices with the same attributes. Logarithmic scale on both axes of both plots.

Figure 1(b) illustrates the effects of basis vector's overlap. The expected overlap of two random vectors is uniquely defined by the number of 1's in basis vectors, i.e., the values in x-axis of Figure 1(b). If basis vectors have high overlap, it becomes harder to distinguish the basis vectors using association confidences. Thus higher overlap degrades the quality of DBP results, as one can clearly see from Figure 1(b). On the other hand, rounded SVD and NMF seem to have more problems on reconstructing data with high overlap in basis vectors. In this experiment, rounded SVD and NMF are again the best, plain SVD and NMF the worst, and DBP is in between. The only exception is the first point, in which the DBP is the best. The explanation is that with small overlap, Algorithm 1 is very effective.

Reconstruction errors for real data. Reconstruction errors for the real datasets are given in Tables 3 (d_1 distance) and 4 (d_2 distance).

We used two additional methods in these experiments, namely *0–1* SVD and *0–1* NMF. The idea is to make binary the factor matrices of SVD and NMF and multiply them using Boolean algebra. However, it is far from obvious how to binarize the factor matrices. We used again a threshold approach: values below

the threshold are rounded to 0 and values above the threshold are rounded to 1. To be fair, we used a brute-force search to select the optimal thresholds for both factor matrices (different threshold for each matrix).

Table 3 shows that in d_1 DBP is comparable to other methods, including rounded SVD. For example, in 20 Newsgroups and NSF Abstracts with $k = 5$ DBP gives the smallest error. While DBP cannot beat SVD or NMF in d_2 (Table 4), it is not too far away from them in most of the cases. The 0–1 SVD and 0–1 NMF have the largest reconstruction error.

Table 3. Reconstruction error of real-world datasets using d_1 distance. Values are scaled and truncated to three decimals.

			algorithm						
dataset	k	scale	SVD	NMF	r. SVD	r. NMF	0–1 SVD	0–1 NMF	DBP
NSF Abstr.	5	10^6	1.124	1.089	0.563	0.563	5.171	3.328	0.559
NSF Abstr.	10		1.152	1.091	0.561	0.561	6.065	5.890	0.554
NSF Abstr.	20		1.197	1.099	0.555	0.556	9.605	10.228	0.545
20 Newsgr.	5	10^6	0.900	0.875	0.450	0.450	4.185	2.612	0.449
20 Newsgr.	10		0.928	0.881	0.446	0.447	4.950	4.447	0.446
20 Newsgr.	20		0.969	0.889	0.440	0.441	7.293	8.096	0.441
Digits	5	10^5	1.308	1.382	0.763	0.855	1.678	1.113	2.133
Digits	10		1.070	1.206	0.471	0.610	1.817	0.967	2.125
Digits	20		0.878	1.028	0.254	0.444	1.678	0.815	2.119
Courses	5	10^4	6.467	6.204	3.215	3.350	8.202	6.418	3.783
Courses	10		6.164	5.779	2.770	2.951	14.051	9.840	3.515
Courses	20		5.949	5.186	2.219	2.495	20.490	17.021	3.160

Empirical time complexity. Set 3 was used to verify the empirical time complexity of the algorithm. The results obtained agreed with theoretical complexity results perfectly, i.e., the running time of Algorithm 1 increased linearly with the number of rows in data and with the size of the basis, and quadratically with the number of columns in data.

Quality of basis vectors for real data. We used the NSF Abstracts dataset to examine the quality of the DBP basis vectors. We used $\tau = 0.3$ and $w^+ = 6$ as the set of parameters that gave the most intuitive results. Examples of basis vectors and representative words are as follows.

`<fund, NSF, year>`,
`<cell, gene, molecular, protein>`,
`<gopher, internet, network, world, wide, web>`,
`<behavior, effect, estim, impact, measure, model, overestimate, predict, test>`, and
`<course, education, enrol, faculty, institute, school, student, undergraduate>`.

Table 4. Reconstruction error of real-world datasets using d_2 distance. Values are rounded to the nearest integer.

		algorithm						
dataset	k	SVD	NMF	r. SVD	r. NMF	0–1 SVD	0–1 NMF	DBP
NSF Abstr.	5	727	728	751	750	2 274	1 825	748
NSF Abstr.	10	719	721	749	749	2 463	2 427	745
NSF Abstr.	20	709	713	745	746	3 099	3 198	738
20 Newsgr.	5	649	650	671	672	2 046	1 616	671
20 Newsgr.	10	643	644	668	669	2 225	2 109	668
20 Newsgr.	20	634	637	664	665	2 701	2 845	665
Digits	5	239	248	276	293	410	334	462
Digits	10	201	221	217	247	426	311	461
Digits	20	168	196	159	211	410	286	460
Courses	5	165	167	179	183	286	253	195
Courses	10	154	158	166	172	375	314	188
Courses	20	141	147	149	158	453	413	178

8 Discussion and Conclusions

We have described the Discrete Basis Problem, investigated its computational complexity, given a simple algorithm for it, and have shown empirical results on the behavior of the algorithm. The results indicate that the algorithm discovers intuitively useful basis vectors. In generated data, the method can reconstruct the basis vectors that were used to generate the data; this holds even with high amounts of noise.

On the other hand, in many cases, SVD has lower reconstruction error than DBP. There are several possible reasons for this. The first possibility is that SVD is in some sense inherently more powerful than DBP. This is of course vaguely expressed. While we know that SVD is optimal with respect to the Frobenius norm, we also know that the Boolean rank of a matrix can be much smaller than its real rank. SVD in some ways has more power than DBP, as SVD works on the continuous values; on the other hand, DBP can take advantage of the Boolean semiring on which it operates. This suggests that the relative performance of DBP against SVD should improve as the overlap between basis vectors increases.

The second alternative reason for the good performance of SVD is that the DBP algorithm is suboptimal. This suboptimality certainly degrades the results: for example, overlap between the basis vectors makes them harder to be discovered. However, for our generated data, in many cases, the DBP algorithm reconstructs the original basis vectors perfectly. Thus, at least for those data sets the algorithm is sufficiently good.

We have shown that Boolean approaches to matrix decomposition form a viable alternative for traditional methods. For further work, it would be of interest to understand the relationship between the approximate Boolean and real ranks

of binary matrices better. Also, a more detailed comparison of DBP against the probabilistic approaches such as LDA and multinomial PCA would be useful.

References

1. Golub, G., Van Loan, C.: Matrix Computations. JHU Press (1996)
2. Lee, D., Seung, H.: Learning the parts of objects by Non-negative Matrix Factorization. Nature **401** (1999) 788–791
3. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. Journal of Machine Learning Research **3** (2003) 993–1022
4. Buntine, W.: Variational extensions to EM and multinomial PCA. In: ECML. (2002)
5. Lee, D., Seung, H.: Algorithms for Non-negative Matrix Factorization. Advances in Neural Information Processing Systems **13** (2001) 556–562
6. Hofmann, T.: Probabilistic Latent Semantic Indexing. In: ACM Conference on Research and Development in Information Retrieval. (1999) 50–57
7. Kabán, A., Bingham, E., Hirsimäki, T.: Learning to read between the lines: The aspect Bernoulli model. In: ICDM. (2004)
8. Seppänen, J., Bingham, E., Mannila, H.: A simple algorithm for topic identification in 0–1 data. In: PKDD. (2003)
9. Koyutürk, M., Grama, A., Ramakrsihnan, N.: Compression, clustering, and pattern discovery in very-high-dimensional discrete-attribute data sets. IEEE Transactions on Knowledge and Data Engineering (2005) 447–461
10. Gionis, A., Mannila, H., Seppänen, J.K.: Geometric and combinatorial tiles in 0-1 data. In: PKDD. (2004)
11. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Discovery Science: 7th International Conference, Proceedings. (2004)
12. Besson, J., Pensa, R., Robardet, C., Boulicaut, J.F.: Constraint-based mining of fault-tolerant patterns from boolean data. In: KDID. (2006)
13. Mishra, N., Ron, D., Swaminathan, R.: On finding large conjunctive clusters. In: COLT. (2003)
14. Brayton, R.K., Hachtel, G.D., Sangiovanni-Vincentelli, A.L.: Multilevel logic synthesis. Proceedings of the IEEE **78**(2) (1990) 264–300
15. Banerjee, A., et al.: A generalized maximum entropy approach to Bregman coclustering and matrix approximations. In: KDD. (2004) 509–514
16. Monson, S.D., Pullman, N.J., Rees, R.: A survey of clique and biclique coverings and factorizations of $(0,1)$-matrices. Bulletin of the ICA **14** (1995) 17–86
17. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
18. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in computer science. Springer-Verlag New York (1999)
19. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD. (1993)
20. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the Twelfth International Conference on Machine Learning. (1995) 331–339
21. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998)

Evaluation of Summarization Schemes for Learning in Streams

Alec Pawling, Nitesh V. Chawla, and Amitabh Chaudhary

University of Notre Dame
{apawling, nchawla, achaudha}@cse.nd.edu

Abstract. Traditional discretization techniques for machine learning, from examples with continuous feature spaces, are not efficient when the data is in the form of a stream from an unknown, possibly changing, distribution. We present a time-and-memory-efficient discretization technique based on computing ε-approximate exponential frequency quantiles, and prove bounds on the worst-case error introduced in computing information entropy in data streams compared to an offline algorithm that has no efficiency constraints. We compare the empirical performance of the technique, using it for feature selection, with (streaming adaptations of) two popular methods of discretization, equal width binning and equal frequency binning, under a variety of streaming scenarios for real and artificial datasets. Our experiments show that ε-approximate exponential frequency quantiles are remarkably consistent in their performance, in contrast to the simple and efficient equal width binning that perform quite well when the streams are from stationary distributions, and quite poorly otherwise.

1 Introduction

Increasing number of real world applications now involve *data streams*; *e.g.*, applications in telecommunications, e-commerce, stock market tickers, fraud and intrusion detection, sensor networks, astronomy, biology, geography, and other sciences [1]. These data streams, whether commercial or scientific, spatial or temporal, almost always contain valuable knowledge, but are simply too fast and too voluminous for it to be discovered by traditional techniques. The objective of modern practitioners is to find time-and-memory-efficient ways of modeling and learning from the streaming data, possibly at the cost of some loss in accuracy. The generally accepted definition of time-and-space efficiency in streams is this: if n elements of the stream have been seen thus far, the algorithm should take time and space at most polylogarithmic in n [1].

There are special cases in which it is relatively easy to learn efficiently in data streams. One such scenario is when the stream is assumed to be a random sample drawn from a stationary or a slowly shifting distribution, often called the "i.i.d. assumption" (for independent and identically distributed). In such situations, a reasonably sized sample, typically not larger than $\mathsf{polylog}(n)$, of the data stream can be assumed to describe the overall distribution (and hence,

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 347–358, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the entire stream) quite accurately, and the problem reduces to learning from such a sample. Another such case is when the underlying domain(s) of the data values is *discrete*, either nominal or consisting of a few numerical values (again, at most $\mathsf{polylog}(n)$). The trick in discrete domains is to simply count the number of instances of each value and use that as a representation of the data. Often this simple representation is sufficient and we get by with memory usage much less than the actual size of the stream.

Many real-world applications, however, involve *continuous* feature values from unknown, and possibly changing, distributions. In these, learning algorithms designed for nominal feature spaces are inapplicable. A simple work-around is to discretize the continuous feature space, and then use techniques similar to those for nominal feature spaces. Discretization of continuous feature spaces is, in fact, quite common in machine learning, primarily because many machine learning algorithms either require a discrete feature space or perform better with discretized features. Naive-Bayes, for example, often uses discretized features to improve performance. Accordingly many supervised and unsupervised discretization methods have been proposed; e.g., Dougherty et al. [2] identify defining characteristics and present an empirical evaluation of several methods. Examples of these methods are equal interval width binning, equal frequency width binning, monothetic contrast criterions (supervised and unsupervised), and k-means clustering. Most of these methods are unsuitable for discretizing data that comes in a stream as they, essentially, require the dataset to be *sorted*, and that cannot be done efficiently in a stream.

In this paper, we study discretization techniques that allow us to learn efficiently in data streams with continuous feature spaces even when generated by unknown, changing (and therefore non-i.i.d.) distributions. As motivating applications we use feature selection and Naive-Bayes classification. Our objective is to find a discretization technique(s) for efficiently performing these two learning tasks on streams, and evaluate the technique(s) both analytically and empirically in "worst-case" scenarios. To differentiate from traditional discretization techniques that are not suited for streams, we shall call those we are interested in *summarization techniques*, emphasizing that their job is to summarize the vast amount of information in a stream. For our applications, as we shall see later (in Section 2), a summarization technique needs to be able to answer the following question: For any feature X_i of the examples in the data stream, any value v_i in the domain of X_i, and any class label y associated with the examples, what is the number of examples seen thus far with a class label y and with values for the feature X_i at most v_i? Naturally no scheme that is time-and-memory-efficient in data streams can answer the question exactly. Thus the quality of a scheme depends on the amount of error in the answer, in particular, on the relationships between the nature of the error, the application, and the type of streaming scenario.

The main summarization technique we study is, what we call, the *ε-approximate exponential frequency quantiles* (denoted by exp in the figures) technique. It is based on maintaining quantiles for each feature-class pair. The ϕ quantile of

n elements, for $\phi \in (0, 1]$ is the $\lceil \phi n \rceil$th smallest element in the set (we assume each element is a number, as in our case). An ε-approximate ϕ quantile is an element that lies in a position between $\lceil \phi n(1-\varepsilon) \rceil$ and $\lceil \phi n(1+\varepsilon) \rceil$ in the sorted data set. In our technique we maintain quantiles for values of $\phi = \alpha^{-i}$ for some constant $\alpha > 1$ and $0 \leq i \leq \lg_\alpha n$; hence the term "exponential frequency" in the name. Gupta and Zane [3] developed efficient techniques for maintaining such quantiles and used it in the context of counting inversions in data streams. We extend the technique to perform feature selection. Towards an analytical evaluation of this technique, we show that its performance for feature selection is exceptionally good by proving worst-case bounds on the error introduced in the computation of information entropy (as required in feature selection) (Section 4). To the best of our knowledge these are the first known error bounds related to (continuous) feature selection in streams. Our empirical evaluation of the technique is quite extensive: first of all we compare its performance with two other popular summarization techniques: the *equal width* summary and the *ε-approximate equal frequency quantiles*. The former is just the equal width binning technique constrained to perform in a stream. The latter is an adaptation of the quantile technique to implement equal frequency binning. We compare these three summarization techniques, and an offline optimal technique that is not constrained by time or memory limitations. We examine their performance and memory usage on large-sized real and artificial data sets, under various streaming scenarios: a stationary distribution (i.i.d), various non-stationary (non-i.i.d.) distributions, and even a distribution that has missing feature values (Section 5).

Related Work

Here, constrained by space, we give only a brief survey of the related research. There has been some research in classification in data streams: either on learning a single classifier [4,5] or a set of classifiers [6]. Hulten et al. [5,7] build decision trees in data streams from nominal domains under the i.i.d. assumption on a stationary or a slowly shifting distribution. They make a clever use of the Hoeffding bounds to decide when a sufficiently large sample has been observed and use that to create their tree. If they observe that the distribution is shifting, they change to a different tree that represents the new data. Gehrke et al. developed BOAT [4], an incremental decision tree algorithm that handles continuous feature spaces, and scales to large datasets, often requiring just 2 scans of the data (we limit ourselves to one).

Gama and Pinto [8] present a method for incremental discretization on a data stream that is based on a histogram summary of the data. They use these histograms to further discretize the data using a variety of the standard discretization techniques (equal width, equal frequency, entropy-based *etc.*) and test the accuracy of Naive-Bayes classifiers built from these discretizations on relatively small datasets (≤ 58000 examples). Guha et al. [9] give a number of methods for computing the entropy of a data stream: a two-pass algorithm that computes a $(1+\varepsilon)$ approximation in $\tilde{O}(1/(\varepsilon^2 H))$ space, where H is the entropy, a single-pass $e/(e-1)+\varepsilon$ approximation algorithm that takes $\mathsf{polylog}(n)$ space, as well as a single pass algorithm that achieves a multiplicative approximation

even when H is very small, but uses significantly more space. It is not clear, however, if these algorithms can be adapted to compute information gain in a data stream.

2 Motivating Applications

We use feature selection and Naive-Bayes classification as our applications. Due to space constraints, we omit the discussion and results of Naive-Bayes; however, these are available in [10]. Both these methods have commonly been studied in the context of discretization techniques even in non-streaming scenarios, as they are both typically implemented using discretized data.

Feature selection is a common process in machine learning, in which a subset of the features available from the data are selected for application of a learning algorithm, such as for classification. Selecting features based on information gain is a popular method; in this a feature is selected based on the maximum decrease in information entropy possible from a partition based on that feature.

Specifically, let there be d features denoted by $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ and c class labels denoted by the set $\mathbf{Y}$. Let S_n denote the stream after n examples. In particular, $S_n = (\mathbf{x}_{(1)}, y_{(1)}), (\mathbf{x}_{(2)}, y_{(2)}), \ldots, (\mathbf{x}_{(n)}, y_{(n)})$ is the stream of n ordered pairs, each consisting of a feature vector $\mathbf{x}_{(i)}$ and a class label $y_{(i)}$. Suppose we wanted to determine the information gain resulting from partitioning the examples in S_n at $X_i = v_i$. Let S_n^L be the subset of examples such that $X_i \leq v_i$ and S_n^R be the remaining examples in S_n. Additionally, let $S_{n,y}$ denote the examples in S_n with the class label y. The information entropy in S_n is defined as

$$I(S_n) = \sum_{y \in \mathbf{Y}} \left(-\frac{|S_{n,y}|}{|S_n|} \lg \frac{|S_{n,y}|}{|S_n|} \right). \tag{1}$$

The information gain by partitioning at $X_i = v_i$ is defined as the decrease in entropy due to the partitioning, i.e.,

$$\text{gain}(S_n, X_i, v_i) = I(S_n) - \left[\frac{|S_n^L|}{|S_n|} I(S_n^L) + \frac{|S_n^R|}{|S_n|} I(S_n^R) \right] \tag{2}$$

Observe that to compute information gain, we need to be able to compute $|S_{n,y}^L|$, i.e., the number of examples with class y such that $X_i \leq v_i$.

3 Summarization Methods

We now describe the three methods of data streams summarizations that we study. We do not explicitly describe the process, but it will be clear from the description how each method can be used to estimate the answer to the basic question our applications need answered: How many examples with class label y are there in the stream thus far with the X_i feature value at most v_i, for any class y, feature X_i and value v_i?

3.1 Equal Width Summary

Equal width binning is a simple, yet popular, discretization method that divides the the range of each feature into equally sized intervals. It can be accomplished in one pass only if the minimum and maximum values for each feature are known ahead of time. Equal width summary is an adaptation of equal width binning for stream that uses the first k examples of the stream to estimate the minimum and maximum values of each feature. This information is used to create a specified number of bins (which is $\mathsf{polylog}(n)$). The first k examples are placed in their respective bins, and all subsequent examples are used to update counts of items in the bins without storing the example.

3.2 Approximate Exponential Frequency Quantile Summary

The approximate exponential frequency quantile summary is based on the approximate quantile structure developed by Gupta and Zane [3]. In it, we maintain ε-approximate ϕ quantiles for $\phi = \alpha^0, \alpha^{-1}, \alpha^{-2}, \ldots$, in which $\alpha = (1 + \varepsilon)$. The quantile values are denoted $Q(\alpha^0), Q(\alpha^{-1}), Q(\alpha^{-2}), \ldots$ respectively. They are maintained using a collection of random samplers. Here we give a very brief, intuitive description of these samplers; please see the paper by Gupta and Zane [3] for details. Suppose we want to maintain the value at rank $n\alpha^{-i}$ (the α^{-i} quantile). We sample each element in the stream with a probability $T\alpha^i/n$, where $T = O(\log n/\varepsilon^2)$ is a parameter, and keep the T smallest values. We *expect* the largest value in the sampler to have rank $n\alpha^{-i}$, as desired. To ensure we have the correct value *with high probability* we just to need to maintain samplers at finer intervals: we can get ε-approximate quantiles for $\phi = \alpha^{-i}$ with probability $1 - 1/n^2$ if we maintain random samplers corresponding to ranks at β^j, $0 \leq j \leq \log_\beta n$, in which $\beta = 1 + \varepsilon/10$ and $T = 8\,\varepsilon^{-2} \ln n$ [3].

We use the samplers above, albeit with the following modification: we maintain α^{-i} quantiles for only values such that $0 \leq \alpha^{-i} \leq 1/2$ — which we call the *left* quantiles; for the rest, we maintain $1 - \alpha^{-i}$ quantiles, for $0 \leq \alpha^{-i} \leq 1/2$ — which we call the *right* quantiles. Suppose we want to find the number of elements with value at most a given v, we first determine whether v "falls" in the left quantiles or in the right quantiles. Then we determine the largest quantile α^{-i} such that $Q(\alpha^{-i}) \leq v$ (or the smallest quantile $1-\alpha^{-i}$ such that $Q(1-\alpha^{-i}) > v$) and use $n\alpha^{-i+0.5}$ (or $n(1 - \alpha^{-i+0.5})$) as a $(1 + \varepsilon)^{1.5}$-approximation of the required number of elements. On the whole we keep $O(\log_\beta n)$ samplers, so the overall space is $O(\log^2 n/\varepsilon^{-3})$. For learning in a stream, one set of left and right quantiles is maintained for each feature-class pair. This allows us to calculate $(1+\varepsilon)^{1.5}$-approximation of the number of elements of class y with values $X_i \leq v_i$ for any class y, any feature X_i, and any value v_i with high probability.

3.3 Approximate Equal Frequency Quantile Summary

The approximate equal frequency quantile summary is like the approximate exponential frequency quantile summary, except instead of maintaining β^i quantiles for different values of i, we keep the quantiles equally spaced — with equal

number of examples between any consecutive pair. For a fair comparison with the other methods, the number of examples between two consecutive quantiles is chosen so that the same number of bins are created. These quantiles too are maintained independently for each feature-class pair.

4 Analytical Evaluation of Exponential Frequency Quantiles

In this section we present our theoretical evaluation of the performance of the exponential-frequency quantiles summarization when used to compute the information entropy at any point in a stream. In particular, suppose that we have seen n examples in the stream, and we are given a feature X_i and a value v_i, and are asked the following question: What would be the information entropy if the set of examples seen thus far were partitioned into two subsets based on the question "Is $X_i \leq v_i$?" We show that we can bound the error introduced if this information entropy was computed using the exponential-frequency quantile summary. As far as we know, this is the first such bound known for *computing entropy in a data stream*, i.e., by an algorithm that makes one pass over the data stream which contains examples that have continuous features and are generated by some evolving, possibly changing, distribution, and such that the algorithm takes at most polylogarithmic space in the number of examples seen thus far. The error bound has a relative (multiplicative) as well as an absolute (additive) component, and is described precisely in the following theorem.

Theorem 1. *Let the information entropy, at any given point in the stream, resulting from partitioning at $X_i \leq v_i$, for a given X_i and v_i be I^*. Let I be the corresponding information entropy computed using exponential-frequency approximate quantile summarization. Then* $(1 - 9\varepsilon)I^* - 9\varepsilon \leq I \leq (1 + 9\varepsilon)I^* + 9\varepsilon$.

Proof. Let the given point in the stream be after n examples and, as before, let S_n be the set of examples seen thus far. Let S_n^L be the subset of examples such that $X_i \leq v_i$ and S_n^R be the remaining examples in S_n. Additionally, let $S_{n,y}$ denote the examples in S_n with the class label y, for every $y \in Y$. Finally, let $S_{n,y}^L$ be the subset of $S_{n,y}$ such that $X_i \leq v_i$ and $S_{n,y}^R$ be the remaining examples in $S_{n,y}$. We saw earlier that the exponential-frequency approximate quantile summarization can be used to compute a $(1 + \varepsilon)^{1.5} = 1 + 1.5\varepsilon + \Theta(\varepsilon^2)$ approximation of $|S_{n,y}^L|$ and $|S_{n,y}^R|$. We shall show how this ensures a bounded approximation for the information entropy. In the rest of the proof we ignore terms that are $O(\varepsilon^2)$ in the interest of a simpler presentation. The expression for the entropy I^* is

$$\frac{|S_n^L|}{|S_n|} I(S_n^L) + \frac{|S_n^R|}{|S_n|} I(S_n^R).$$

We first develop the upper bound; a lower bound will follow in a similar fashion. For the upper bound, consider $I(S_n^L)$. Observe that this is essentially a sum of terms of the following form, one for each (non-empty) class $y \in Y$:

$$-\frac{|S^L_{n,y}|}{|S^L_n|}\lg\frac{|S^L_{n,y}|}{|S^L_n|}.$$

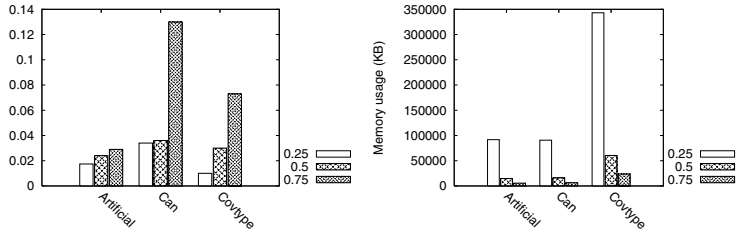
Now $|S^L_n| = \sum_y |S^L_{n,y}|$. Since we can compute an $(1+1.5\varepsilon)$-approximation of $|S^L_{n,y}|$, for each y, it follows that by adding each such approximation we can compute a $(1+1.5\varepsilon)$-approximation of the sum $|S^L_n|$. Thus our approximation of $|S^L_{n,y}|/|S^L_n|$ is upper bounded by $|S^L_{n,y}|(1+1.5\varepsilon)/(|S^L_n|(1-1.5\varepsilon)) = |S^L_{n,y}|/|S^L_n| \cdot (1+3\varepsilon)$, and lower bounded by $|S^L_{n,y}|/|S^L_n| \cdot (1-3\varepsilon)$. Let $|S^L_{n,y}|/|S^L_n|$ be denoted by p, and let the approximate value we obtain be denoted by $\tilde{p}$. Thus we have

$$-\tilde{p}\lg\tilde{p} \le -p(1+3\varepsilon)\lg(p(1-3\varepsilon)).$$

(Recall that $-\lg\tilde{p}$ is a positive number since $\tilde{p}$ is always at most one, and that the right hand side has the factor $-\lg(p(1-3\varepsilon))$ instead of $-\lg(p(1+3\varepsilon))$ since it is the larger of the two.) We consider two cases: the first will result in both relative and absolute error terms, and the second in just a relative error term. In Case 1, assume $p > 1/2$. If we use the approximation $\ln(1-x) = -x$ and ignore terms that are $O(\varepsilon^2)$, we get the following bound: $-\lg(p(1-3\varepsilon)) = -\lg p - \ln(1-3\varepsilon)/\ln 2 = -\lg p + 3\varepsilon/\ln 2 \le -\lg p + 4.33\varepsilon$. Thus $-\tilde{p}\lg\tilde{p} \le -(1+3\varepsilon)p\lg p + (1+3\varepsilon)p \cdot 4.33\varepsilon \le -(1+3\varepsilon)p\lg p + 4.33\varepsilon$, since $p \le 1$. For Case 2, assume $p \le 1/2$. Thus $-\lg(p(1-3\varepsilon)) \le -\lg p + 4.33\varepsilon$ as before, which in this case is bounded by $-\lg p(1+4.33\varepsilon)$ since $-\lg p \ge 1$. This results in $-\tilde{p}\lg\tilde{p} \le -(1+3\varepsilon)p(1+4.33\varepsilon)\lg p = -(1+7.33\varepsilon)p\lg p$. Note, in $I(S^L_n)$, at most one such term can have $p > 1/2$, i.e., be in Case 1. Thus, by adding the approximations for each term, we have an approximation for $I(S^L_n)$ that is upper bounded by $(1+7.33\varepsilon)I(S^L_n)+4.33\varepsilon$. To bound the term $|S^L_n|/|S_n| \cdot I(S^L_n)$ in our expression for entropy, note that $|S_n| = n$ is known exactly, and we can obtain an $(1+1.5\varepsilon)$-approximation for $|S^L_n|$. We can similarly bound the approximation we get for $|S^R_n|/|S_n| \cdot I(S^R_n)$. Combining both, we get the following upper bound on the computed value of the entropy I: $(1+1.5\varepsilon)(1+7.33\varepsilon)I^* + 2\cdot(1+1.5\varepsilon)\cdot 4.33\varepsilon \le (1+8.53\varepsilon)I^* + 8.66\varepsilon$.

5 Empirical Evaluation

In this section we evaluate the accuracy and memory usage of the three summarizations empirically for feature selection and compare them with an offline algorithm that has no time or memory constraints. We present results only for experiments in which ε (as required by the approximate quantiles) is set to 0.5. This is because, based on experiments presented in [11] (e.g., see Figure 1), at this value we have, informally, a good compromise between low relative error and reasonable usage of memory. In experiments using equal width summaries we use $k = 100$ examples to establish the bin boundaries. In addition, for a fair comparison, we maintain as many bins as there are quantiles in the summaries for $\varepsilon = 0.5$.



(a) Relative Error **(b)** Memory Usage

Fig. 1. Computing information gain for different ε: 0.25, 0.50, and 0.75 [11]

Datasets

Large data sets are essential for our experiments, but we were limited to those available in the public domain. Table 1 summarizes the datasets we used. Among the large real data sets we found, we chose four to get a set with different characteristics. Since our method is for continuous features and nominal features will, in practice, be handled separately, we pruned the nominal features (if any) from each dataset. We also generated a large artificial dataset to stress-test the summarizations. Brief description of the data sets: (1) We generated the feature values of the *Artificial* dataset using normal distributions, and added random noise using different normal distributions. The class labels were determined by partitioning the space of the two "important" features into two using a random vector. (2) The *Can* dataset is from a scientific simulation of a can being crushed [12]. The data portion in which the can is being crushed (8,360 out of 443,872 examples) is labeled "very interesting" and the rest labeled "unknown." (3) The *Covtype* dataset is from the UCI repository [13] and is about forest cover types for different geographical features. It has 7 classes with a highly imbalanced distribution; we selected the 10 continuous features out of a total 54 features. (4) The *Census-Income* dataset is also from UCI and highly unbalanced. We selected the seven continuous features out of a total 45 features. (5) The *UCSD* dataset [14] is a large classification dataset used at the UCSD student data mining competition. We pruned its 42 features to 8 in our experiments.

Streaming Scenarios

We tested the summarization techniques on four different types of streaming scenarios: (1) The *Stationary distribution* which models an i.i.d scenario. For this the examples are randomly shuffled to create the stream. (2) The *Non-Stationary–Feature distribution*, in which the examples are sorted in increasing order of the most significant feature to create a "worst-case" non-i.i.d. distribution. (3) The *Non-Stationary–Class distribution*, in which the stream starts with a single class, followed by all classes occurring with the same frequency until all the examples of one (or more) class(s) are exhausted, to create another non-i.i.d. distribution. (4) The *Missing Values distribution*, which is like the stationary

Table 1. Datasets

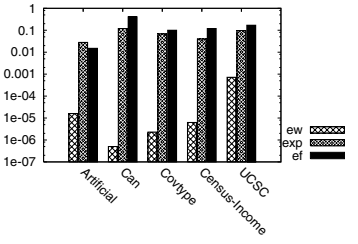
Dataset	Size	Classes	Features
Artificial	1,000,000	2	6
Can	443,872	2	9
Covtype	581,012	7	10
Census-Income	199,523	2	7
UCSD	1,837,583	2	8

distribution, except 30% of the feature values have been removed at random. The non-stationary distributions may seem contrived, but our objective is to develop "bullet-proof" methods for unknown, rapidly evolving, streams.

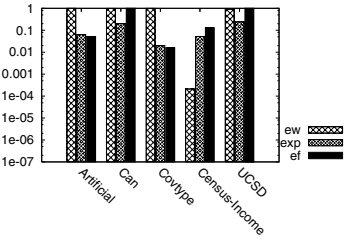
5.1 Performance of Feature Selection

For each dataset and streaming scenario, we calculate the information gain at every tenth point of the total stream. Figure 2 shows the average relative error computing the gain for the best feature (according to gain), plotted for each combination of summarization, dataset, and streaming scenario. The approximate exponential frequency quantile technique shows remarkable consistency: its relative error is almost always between 0.01 and 0.1. When some features have missing values, it performs even better: relative error between 0.001 and 0.01. The likely reason for this is that we let the technique take memory based on the original size of the stream, without accounting for the smaller number of examples for each class-feature pair. In contrast, the equal width summary performs much better than the exponential frequency quantile summary for the stationary, non-stationary–class, and missing values distributions: with error better 10^{-7} and 0.01 most of the time. But for non-stationary–feature distribution, and the Covtype dataset on non-stationary–class distribution, its error shoots up to nearly 1. Thus the exponential frequency quantile summary is better suited for unknown streams. The equal frequency quantile summary rarely performs better than the exponential frequency quantile summary, and performs poorly on the non-stationary–feature distribution. Thus the strength in the exponential frequency quantile summary is in the exponential frequency part of the technique; as was also demonstrated by the analysis in Section 4.

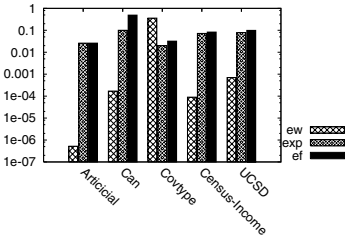
Since we are considering information gain as a tool for feature selection, we examine how the feature rankings for each approximation correlates with the precise feature ranking. Figures 3 shows the *Spearman rank* correlations of the features sorted in descending order of importance, based on the information gain, for the Covtype dataset on the stationary, non-stationary–feature, and non-stationary–class distributions (we only show one dataset due to space constraints). Notice that the equal width summary produces a perfect ranking for the stationary distribution, but degrades on the non-stationary–feature distribution. For the non-stationary–feature distribution, the feature on which the data stream was sorted (in this case, the most important feature) is likely to be



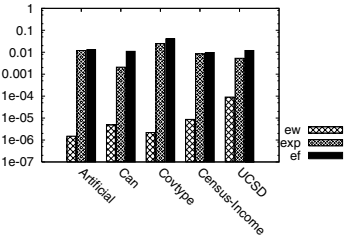
(a) Stationary Distribution



(b) Non-Stationary–Feature Distribution



(c) Non-Stationary–Class Distribution



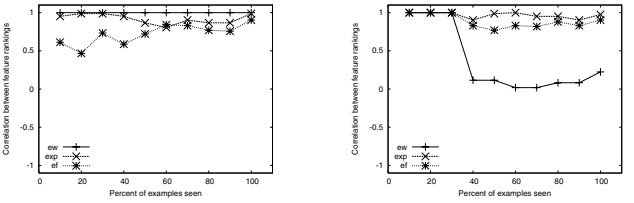
(d) Missing Values Distribution

Fig. 2. Average relative error between the information gain of the best feature for each dataset and streaming scenario; Y-axes are in log-scale

misranked since the bins for that feature are established on a small subset of the range of the feature.

5.2 Memory Usage

Figures 4 shows the memory usage for the offline and three approximations. In all cases, the equal width method needs the least amount of memory. The quantile based summarizations always require less memory than offline algorithm. The anomalous behavior of equal frequency quantiles can be explained. Note that the number of samplers that are actually useful depends upon the number of values that have been inserted into the quantile structure. If the structure isn't full (which it won't be unless all examples belong to a single class) there will be samplers that have been partially populated but aren't used. This is wasted space. This is particularly problematic for the Covtype dataset since there are a large number of classes compared to the number of examples, and some classes have only a small number of examples. If we measure the memory usage where each example in the Covtype dataset is inserted into the quantile summary structures four times — essentially simulating a longer stream — the equal frequency quantile summary requires less memory than offline.



(a) Stationary Distribution **(b)** Non-Stationary–Feature Distribution

Fig. 3. Correlation between the feature ranking of offline and each summarization on Covtype for stationary and non-stationary distributions

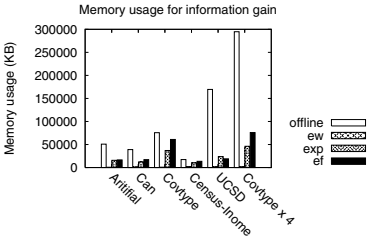


Fig. 4. Memory usage form information gain

6 Conclusion

In this paper we developed the ε-approximate exponential frequency quantiles based technique for summarizing data streams. Although this technique takes time and memory that are at most polylogarithmic in size of the stream, we showed that its performance in computing information entropy is very good by proving worst-case bounds on the error introduced in the computation compared to an offline optimal algorithm that has no time and memory constraints. In addition we empirically evaluated this technique and two other popular techniques — equal width summary and ε-approximate equal frequency quantiles — for feature selection under a variety of streaming scenarios for both real and artificial datasets. The ε-approximate exponential frequency quantiles based technique is remarkably consistent, even in "worst-case" scenarios (its relative error is almost always around 0.1), and should be the technique of choice for streams with continuous feature spaces that are from unknown or non-stationary distributions (when the i.i.d. assumption does not hold). If, however, we know that the i.i.d. assumption is valid for a given stream, the technique to use (not surprisingly) is equal width summary, for its simplicity, performance, and memory usage.

References

1. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proc. 21st ACM Symposium on Principles of Database Systems (PODS). (2002) 1–16
2. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In Prieditis, A., Russell, S., eds.: Proc. 12th International Conference on Machine Learning (ICML), Morgan Kaufmann (1995) 194–202
3. Gupta, A., Zane, F.X.: Counting inversions in lists. In: Proc. 14th ACM-SIAM Symposium on Discrete algorithms (SODA). (2003) 253–254
4. Gehrke, J., Ganti, V., Ramakrishnan, R., Loh, W.Y.: BOAT — optimistic decision tree construction. In: Proc. ACM SIGMOD International Conference on Management of Data. (1999) 169–180
5. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proc. 6th International Conference on Knowledge Discovery and Data Mining (KDD), ACM Press (2000) 71–80
6. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), New York, NY, USA, ACM Press (2001) 377–382
7. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), San Francisco, CA, ACM Press (2001) 97–106
8. Gama, J., Pinto, C.: Discretization from data streams: Applications to histograms and data mining. In: 2nd International Workshop on Knowledge Discovery from Data Streams. (2005)
9. Guha, S., McGregor, A., Venkatasubramanian, S.: Streaming and sublinear approximation of entropy and information distances. In: Proc. ACM SIAM Symposium on Discrete Algorithms (SODA). (2006) 733–742
10. Pawling, A., Chawla, N.V., Chaudhary, A.: Evaluation of summarization schemes for learning in streams. Technical Report 2006-08, University of Notre Dame (2006) `www.cse.nd.edu/research/tech_reports`.
11. Pawling, A., Chawla, N.V., Chaudhary, A.: Computing information gain in data streams. In: ICDM Workshop on Temporal Data Mining: Algorithms, Theory, and Applications. (2005)
12. Chawla, N.V., Hall, L.O.: Modifying MUSTAFA to capture salient data. Technical report, University of South Florida, Computer Science and Engineering Department (1999)
13. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html (1998)
14. UCSD: UCSD student data mining competition. `http://mill.ucsd.edu/ (2006)`

Efficient Mining of Correlation Patterns in Spatial Point Data

Marko Salmenkivi

Helsinki Institute for Information Technology, Basic Research Unit
Department of Computer Science, P.O. Box 68, FI–University of Helsinki, Finland
marko.salmenkivi@cs.helsinki.fi

Abstract. We address the problem of analyzing spatial correlation between event types in large point data sets. Collocation rules are unsatisfactory, when confidence is not a sufficiently accurate interestingness measure, and Monte Carlo testing is infeasible, when the number of event types is large. We introduce an algorithm for mining correlation patterns, based on a non-parametric bootstrap test that, however, avoids the actual resampling by scanning each point and its distances to the events in the neighbourhood. As a real data set we analyze a large place name data set, the set of event types consisting of different linguistic features that appear in the place names. Experimental results show that the algorithm can be applied to large data sets with hundreds of event types.

1 Introduction

Consider a large set of spatial point objects. Each object may be an instance (*event*) of zero, one, or several event types of interest, and the number of event types is large (e.g, 100-2000). We are interested in analyzing spatial correlation between the point patterns of different event types.

As an example of real data, Fig. 1 illustrates the variation of frequency of place names in East Finland in a corpus of the National Land Survey of Finland. A small dot is plotted at all named locations. For illustration purposes, the events of three linguistic features occurring in the place names are also indicated. The indicated event types are some of the lexical features that are of interest in the study of the settlement in East Finland during the Iron Age, and the ancient interaction between the Finns and the Saami in the region.

These features are just examples. As a very large set of different lexemes, and other linguistic features, appear in place names, the number of event types may be large, hundreds, or even more.

Fig. 1 shows that the overall frequency of place names is not constant across the area. We see, for instance, that lakes are discerned from the surroundings with lower frequencies of events, whereas on the shores the density of events is high. Instances of event types may correlate simply because they both occur more frequently in the regions of high overall intensity, and less frequently in the regions of low intensity. Thus, to obtain reliable results, we desire to relate the interestingness of an observed correlation to the overall frequency of objects.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 359–370, 2006.
© Springer-Verlag Berlin Heidelberg 2006

There are no methods available that can do with a lot of data and more than a few event types. Collocation rules only consider the plain frequencies of nearby instances of event types in each rule, ignoring the overall frequency of events. They can include several event types, but in practice the algorithms may be very slow if the number of event types is large [9]. In spatial statistics Monte Carlo tests are usually employed. They are computationally infeasible when the number of event types is large.

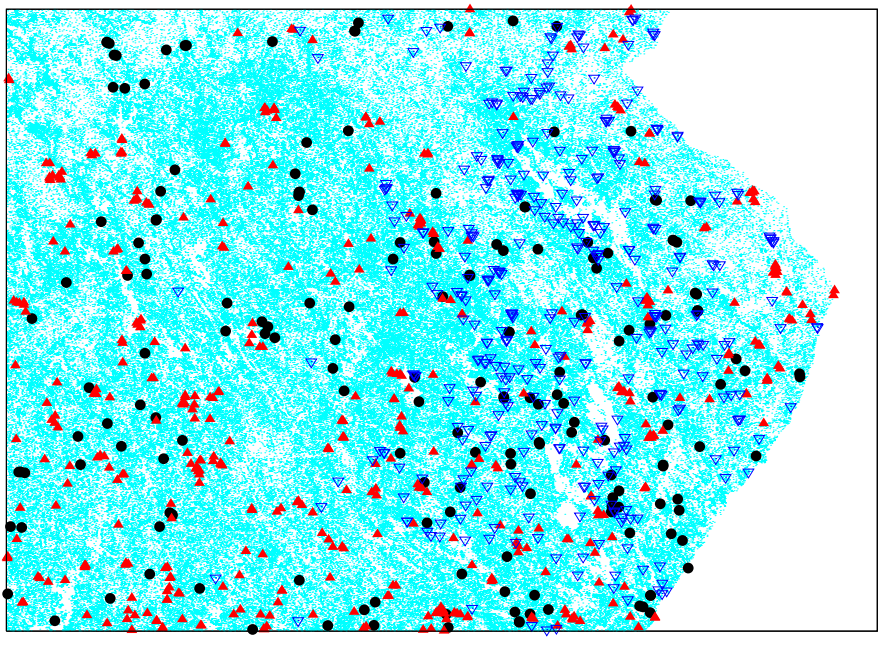


Fig. 1. Locations of place names (very small dots) in East Finland in the corpus of the National Land Survey. Instances of three event types: 1) unfilled (blue) triangles =name elements assumed to be of Saami origin; 2) filled (red) triangles = instances of lexemes *akka,akko*, 'old woman'; 3) circles = instances of lexeme *louhi* (the female ruler of *Pohjola* in *Kalevala*, the Finnish national epic.)

We propose a method that can be seen as a non-parametric bootstrap test that, however, avoids the actual resampling from the data by scanning each point and its distances to the points in the neighbourhood in turn. A preliminary version of the test, using Monte Carlo methods, was outlined in [6]. In this paper we extend the idea to large sets of event types by introducing a novel algorithm that computes the tests without actual resampling. Theoretical complexity of the algorithm is $\mathcal{O}(n^2)$, where n is the number of events. In practice, a small fraction of distances between points has to be evaluated, and our experiments show that the algorithm can evaluate correlations in large data sets and event type sets in feasible time. The results provide more detailed knowledge than

collocation rules, since the distribution of all points is considered when assessing how interesting individual rules are.

This paper is organized as follows. In Section 2 the related work is introduced. In Section 3 the notion of correlation pattern is proposed for measuring the significance of two-feature collocation rules. A new algorithm for finding such patterns is introduced in Section 4. Experiment results on real and synthetic data sets are provided in Section 5. Section 6 is a conclusion.

2 Related Work

Collocation rules are often used to describe dependencies in spatial data. Collocation rule mining algorithms are typically based on first finding interesting *collocation patterns*, and then extracting rules from them. A neighbourhood relation R of spatial objects is assumed to be explicitly given. In case of points, the definition is usually based on the Euclidean distance of at most a predefined ϵ Let $\mathcal{E}$ be a set of features (or *event types*). A collocation pattern is defined as a set $\mathcal{Q} \subseteq \mathcal{E}$ [3,4,8,9].

Further let $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be a set of spatial point objects. The type of data considered in this paper can be represented as a binary $S \times \mathcal{E}$-matrix. We say that $p \in S$ is an instance (or *event*) of event type $A \in \mathcal{E}$ in data D if $D_{p,A} = 1$. A set S' of spatial objects is a *(row) instance* of $\mathcal{Q}$ if for all $o_i, o_j \in S', dist(o_i, o_j) \leq \epsilon$, and S' contains instances of all the event types in $\mathcal{Q}$, and no proper subset of S' does so [3].

Let $\mathcal{P}$ and $\mathcal{Q}$ be collocation patterns, and $\mathcal{P} \cap \mathcal{Q} = \emptyset$. Then $\mathcal{P} \rightarrow \mathcal{Q}$ is a *collocation rule*. The *confidence* of $\mathcal{P} \rightarrow \mathcal{Q}$ in data D is the fraction of instances of $\mathcal{P}$ such that they are also instances of $\mathcal{P} \cup \mathcal{Q}$.

Interestingness measures for collocation patterns include *prevalence*, which is defined as $\min\{pr(F, \mathcal{P}), F \in \mathcal{P}\}$, and *maximum participation ratio* (MPR), $\max\{pr(F, \mathcal{P}), F \in \mathcal{P}\}$, where $pr(F, \mathcal{P})$ is the proportion of the events of type F that appear in the instances of $\mathcal{P}$. A high prevalence indicates that $\mathcal{P}$ can be used to generate confident collocation rules [8]. Correspondingly, a high MPR implies that at least one of the event types, denote it by T, rarely occurs outside $\mathcal{P}$. Hence, the collocation rule $\{T\} \rightarrow \mathcal{P} \setminus \{T\}$ is likely to be interesting [4].

Levelwise search algorithms for finding all interesting patterns were introduced by [4,8]. The algorithm proposed by Zhang et al. finds for each object in turn the maximal pattern instance in which the object participates [9].

Let $Z \subset \mathbf{R^2}$ be the observation region under investigation. In spatial statistics point patterns are typically modeled as point processes $\{X(\mathbf{t}) : \mathbf{t} \in Z\}$ which generate events $\mathbf{t}$ on Z. If there are k event types, $X(\mathbf{t})$ is a k-dimensional binary vector indicating the event types whose instance point $\mathbf{t}$ is. Hypothesis testing selects a point process for modeling the null hypothesis H_0 of no spatial correlation, and a test statistic T measuring the correlation in an observed point pattern. The simplest choice is the Poisson process conditioned on the numbers of events of the event types to be tested (complete spatial randomness, CSR). In practice, Monte Carlo methods are usually needed to simulate the selected process to obtain an approximation for the sample distribution of T [1,2].

Intensity function of a point process intuitively describes the average frequency of events in a unit area. It is defined as $\lambda(\mathbf{s}) = \lim_{|ds|\to 0} E(Y(ds))/|ds|$, where ds is a "small" region around $\mathbf{s} \in Z$, $|ds|$ is the area of ds, and $Y(ds)$ is the random variable indicating the number of events in ds. For the number of instances in $Z' \subseteq Z$, it holds that $E(Y(Z')) = \int_{Z'} \lambda(\mathbf{s})ds$.

Leino et al. study significances of two-feature collocation rules in lake name data, where different lake names form the set of event types. Their approach takes the overall frequency of lakes into account, but unrealistically assumes the probability of an instance of each event type to be constant across Z [5].

3 Testing Collocation Rules

We next consider collocation rules of type $A \to B$, where A and B are single event types, in the context of hypothesis testing. The notation is summarized in Table 1.

Let us assume that the data S (or $S \times \mathcal{E}$) is a sample from population F. It is natural to investigate a test statistic based on the definition of confidence of a collocation rule. Accordingly, let us denote the conditional probability of observing an instance of B within a distance $\leq \epsilon$ from a given instance of A by $G_{A,B}(\epsilon)$, and its observed value, which is the confidence of $A \to B$, in data by $g_{A,B}(\epsilon)$. We define a *correlation pattern* as a statistically significant collocation rule $A \to B$ as follows:

Definition 1. *Let $\{X(s) : s \in Z\}$ be a bivariate spatial point process under a null hypothesis H_0, that is, it specifies the probability distribution of point patterns of two event types, denote them by A and B, under H_0. A collocation rule $A \to B$ is a correlation pattern in data $S \times \mathcal{E}$ with significance level α iff $Prob(G_{A,B}(\epsilon) \leq g_{A,B}(\epsilon)|X(s)) \leq \alpha$.*

For practical purposes, the general definition is, of course, useful only if a computationally feasible H_0 can be found, and it describes in a meaningful way the vague concept of "no spatial correlation" between A and B. Below we devise methods to solve this key problem.

The CSR null hypothesis that assumes a constant intensity of the generating process everywhere in Z is too simple in practice. Assuming that, in addition to events of types A and B, a lot of point data is available, the following bootstrap test may be a better alternative: sample from S to obtain A^* and B^*, and compare $g_{A,B}(\epsilon)$ against the obtained distribution of $\hat{G}_{A,B}(\epsilon)|H_0$. Now, the intensity of the generating process under H_0 is not assumed to be constant but a bootstap estimate $\lambda_{\hat{F}}(\mathbf{s})$ of λ in population F.

This testing procedure works, if the intensities λ_A and λ_B can be assumed to be proportional to λ_F. Unfortunately, this assumption is often not valid either. For the same reason permutation testing of the whole data is not meaningful. All event types do not usually occur in the whole area of investigation, and they may be rare somewhere, and common somewhere else, not necessarily following

Table 1. Notation

Symbol	Explanation
$\mathcal{E} = \{A, B, \ldots\}$	set of event types
S	set of all spatial point objects in current data
F	population from which S is a sample
$\mathcal{A}, \mathcal{B}, \ldots$	sets of events of event type A, B etc.
$G_{A,B}(\epsilon)$	conditional prob. in F: given $a \in \mathcal{A}, \exists b \in \mathcal{B}, s.t. dist(a,b) \leq \epsilon$.
$g_{A,B}(\epsilon)$	observed value of $G_{A,B}(\epsilon)$ in current data
$\lambda_A(\mathbf{s})$	intensity of the process (at location $\mathbf{s}$) that generated $\mathcal{A}$
$\mathcal{A}$	pseudosample of A (number of events equal to $\lvert\mathcal{A}\rvert$)

the variation of frequency of all objects (see, e.g., the event type indicated by unfilled triangles in Fig. 1).

Thus, to be realistic H_0 should somehow depend on the spatial variation of frequencies of A and B, of course in a carefully controlled way. Accordingly, consider generating events independently for A^* and B^* from the processes whose intensities are kernel density estimates $\hat{\lambda}_A(\mathbf{s})$ and $\hat{\lambda}_B(\mathbf{s})$. A kernel density estimate $\hat{\lambda}_A$ is obtained by inserting, e.g., a bivariate Gaussian f, around each $a_i \in \mathcal{A}$, and $\hat{\lambda}_A(\mathbf{s}) = \sum_{i=1}^{n} \frac{1}{h^2} f(\frac{\mathbf{s}-a_i}{h})$. The parameter h (*bandwidth*) controls the degree of smoothing.

When $h \to \infty$, this corresponds to sampling under the CSR. On the other hand, when $h \to 0$ the procedure converges to the resampling from sets $\mathcal{A}$ and $\mathcal{B}$. This is the regular bootstrap approach for parameter estimation, but it is not a valid approach for testing, since the resamples are not drawn under H_0. However, by conducting trials with several h we can study the value of the test statistic as the function of h, that is, the impact of "weighting" the CSR hypothesis by $\hat{\lambda}_A(\mathbf{s})$ and $\hat{\lambda}_B(\mathbf{s})$.

While this method is interesting, it still investigates only the events of types A and B, ignoring the information available in the distribution of S. A simple modification that yields a better solution is as follows. Consider sampling from S to obtain A^* and B^*. Let us weight each point $\mathbf{t} \in S$ by $\hat{\lambda}_A(\mathbf{t})$ (or $\hat{\lambda}_B(\mathbf{t})$). When $h \to \infty$, the same weight is assigned to all points, resulting in plain sampling from S. Similarly to the previous case, when $h \to 0$ the procedure converges to resampling from sets $\mathcal{A}$ and $\mathcal{B}$.

Fig. 2 illustrates this in case of two event types in the place name data (lexemes *musta* 'black' and *valkoinen* 'white'). The indicated different bandwidths correspond to different weightings of the plain sampling from S, which is titled "no weight". The errorbars show the 99 % intervals of $\hat{G}_{A,B}(\epsilon)|H_0$. The results show that the observed $g_{A,B}(1\text{ km}) = 275$ cannot be obtained even by the strongly weighted null hypotheses ($h = 0.5$ km), and, hence, the rule $A \to B$ is clearly very significant. The plain sampling from S indicates remarkable deviation from H_0, when the neighbourhood is extended to 4 km, whereas $g_{A,B}(4$ km) is included in the 99 %-intervals of more realistic weighted versions.

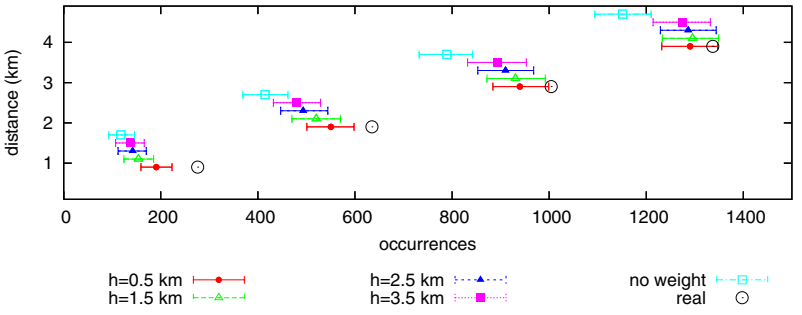


Fig. 2. Significance of rule $A \to B$ with different neighbourhood definitions (distance in y-axes), and weighted versions of H_0 (bandwidth h of Gaussian kernel). Errorbars and circles indicate 99 %-intervals of $\hat{G}_{A,B}(distance)|H_0$, and $g_{A,B}(distance)$, respectively.

All the described testing procedures rely on Monte Carlo methods. We next propose a method that avoids them. Consider the previous approach, but fix the events in $\mathcal{B}$, and only draw pseudosamples A^* from S. If $|S|$ is small, or no reference data is available, i.e. $S = \mathcal{A} \cup \mathcal{B}$, then it is easy to see that $E(\hat{G}_{A,B}|H_0) > g_{A,B}$, i.e., the pseudosamples are on average closer to some $b \in \mathcal{B}$ than the original data, and thus, the estimator is biased. Obviously, when more reference data are available, the dependence becomes weaker, and in the limit, when $|S| \to \infty$, $|\mathcal{A}|/|\mathcal{S}| \to 0$, and $|\mathcal{B}|/|\mathcal{S}| \to 0$, then $E(\hat{G}_{A,B}|H_0) \to G_{A,B}|H_0$. Thus, the estimator is asymptotically unbiased. Results on synthetic data in Sec. 5.1 demonstrate this.

Conditioning the method on one of the event types makes it possible to develop an efficient algorithm for computing the significant associations without actually generating the pseudosamples. The assumptions needed are, accordingly, that S is "large", and the proportions $|\mathcal{A}|/|\mathcal{S}|, |\mathcal{B}|/|\mathcal{S}|$ of individual event types to be tested are "small".

Let us again consider rule $A \to B$. Now, it holds, under H_0, that $G_{A,B}(\epsilon) \sim \text{Bin}(|\mathcal{A}|, p_{A,B})$, where $p_{A,B}$ is the probability that during the resampling for A^* an event that is within ϵ from an event of type B is selected. The probability p is determined by the kernel estimates $\hat{\lambda}_A(\mathbf{t}), \mathbf{t} \in S$.

Denote by $S^* \subseteq S$ the set of events such that for each $x \in S^*, dist(x, b) \leq \epsilon$ for some $b \in \mathcal{B}$. Then,

$$p_{A,B} = \frac{\sum_{\mathbf{x} \in S^*} \hat{\lambda}_A(\mathbf{x})}{\sum_{\mathbf{y} \in S} \hat{\lambda}_A(\mathbf{y})} = \frac{\sum_{\mathbf{x} \in S^*} \sum_{\mathbf{a_i} \in \mathcal{A}} f(\frac{\mathbf{x}-\mathbf{a_i}}{h})}{\sum_{\mathbf{y} \in S} \sum_{\mathbf{a_i} \in \mathcal{A}} f(\frac{\mathbf{y}-\mathbf{a_i}}{h})}. \quad (1)$$

The sums in Eq. 1 can be approximately computed by inspecting the occurrences "sufficiently" close to the events in $\mathcal{A} \cup \mathcal{B}$. The significance of the observed number of co-occurrences can then be evaluated by comparing $g_{A,B}(\epsilon)$ with the binomial distribution of $G_{A,B}|H_0$ (e.g., by applying normal approximation).

4 Algorithm for Finding Associations

In this section we introduce an algorithm that computes the significances of correlation patterns between the instances of pairs of event types.

A useful kernel function decreases monotonically as a function of distance from its mode. For instance, 99.7% of the probability mass of a bivariate Gaussian kernel function f with bandwidth h is concentrated inside a circle with radius $3h$. Hence, by setting, for example $\eta = 4h$, we can safely assume that $f(\eta) \approx 0$, and ignore the events that are not within η from the mode. When inspecting each $\mathbf{t} \in S$ to compute the approximations for $p_{X,Y}, (X, Y) \in \mathcal{E} \times \mathcal{E}$ in Eq. 1, we use this: the "influence" of $\mathbf{t}$ does not extend to any location $\mathbf{s}$ such that $dist(\mathbf{s}, \mathbf{t}) > \eta$.

Consider location $\mathbf{t}$ in the left panel of Fig. 3. The events of type A, and the closest event of type B (b_1) are shown. The events a_4, a_5 that are not within a distance of η from $\mathbf{t}$ are indicated by dashed edges. One should compute $\hat{\lambda}_A(\mathbf{t})$, i.e., to evaluate the kernel function three times to obtain $w = \sum_{i=1}^{3} f(\frac{\mathbf{t}-\mathbf{a_i}}{h})$. This yields the contribution of $\mathbf{t}$ to the denominator of Eq. 1. Further, if the distance of $\mathbf{t}$ and b_1 is less than ϵ, w should also be included in the sum of the numerator in Eq. 1.

The right panel of Fig. 3 describes a more realistic setting, where there are several event types, and one should compute $p_{i,j}$ for each ordered pair of event types $(i, j) \in \mathcal{E} \times \mathcal{E}$. Now all the events within η from the central node $\mathbf{t}$ are shown: $a_2 \in \mathcal{A}, b_1, b_2 \in \mathcal{B}$, and $c_1, c_2 \in \mathcal{C}$. The events (b_1, c_2) within a distance less than ϵ from $\mathbf{t}$ are indicated by thick edges. Instead of only one pair of event types, we now have to find the sums for all pairs $(i, j) \in \{A, B, C\}, i \neq j$, according to exactly the same simple procedure as in the previous case.

The procedure can be generalized as follows. Let $\mathbf{t}$ be the location of the central object, and further let y be an instance of Y such that $dist(\mathbf{t}, y) \leq \epsilon$. If there are instances $x_1, x_2, \ldots, x_k$ of X such that $dist(\mathbf{t}, x_i) < \eta, 1 \leq i \leq k$ then, for computation of $p_{X,Y}$ of Eq. 1, the term $\sum_{i=1}^{k} f(\frac{\mathbf{t}-x_i}{h})$ has to be included in the numerator of Eq. 1. If several event types are allowed to occur at one location, the sums for each event type are updated at each location.

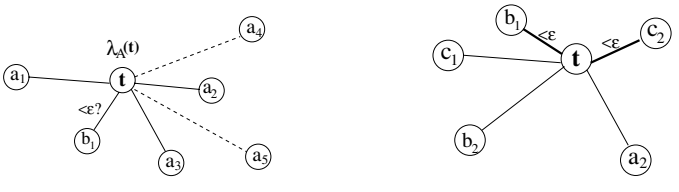


Fig. 3. Star graphs for computing sums of kernel density estimates

Hence, consider the following algorithm for testing collocation rules. The objects are first sorted according to their x-coordinates in order to avoid computing distances between all objects (line 1). The y-coordinates of two objects are inspected only if their x-coordinates are sufficiently close to each other, and the

kernel function is evaluated only if the distance of objects $< \eta$ (lines 7–11). The main loop (lines 3–19) scans the η-neighbourhood of each object. Lines 4–6 initialize the list L of features with an instance within η from o_i, and the sums of weights of each event type. Each event type whose instance is closer than η to o_i is added to L in line 13. Indicator function $f_X(o_i)$ returns TRUE if o_i is an instance of event type X (lines 12–13). The sums of each pair of event types are updated in function **update_pairwise_weights**. The counter of real co-occurrences of event types X and Y is $g[X, Y]$. These counters are updated in line 18. After iterating all objects in S, the significances of $g[X, Y]$ are evaluated by comparing them to the binomial distributions of the test statistics.

Algorithm. **test_collocations** $(o_1, \ldots, o_n \in S, \epsilon, \eta,$ set of event types $\mathcal{E})$

1. sort $o_1, \ldots, o_n$ according to their x-coordinates;
2. $k \leftarrow 1$
3. **for** each $1 \leq i \leq n$ do
4. $\quad L \leftarrow \emptyset$
5. $\quad$ **for** each $X \in \mathcal{E}$ **do**
6. $\quad\quad w_\epsilon[X] \leftarrow 0;\ w_\eta[X] \leftarrow 0$
7. $\quad$ **while** $o_i.x - o_k.x \geq \eta$ **do** $k \leftarrow k + 1$
8. $\quad j \leftarrow k$
9. $\quad$ **while** $o_i.x - o_j.x < \eta$ **do**
10. $\quad\quad$ **if** $(dist(o_i, o_j) < \eta)$ **then**
11. $\quad\quad\quad z = weight(o_i, o_j)$
12. $\quad\quad\quad$ **for** each $\{X \in \mathcal{E} | f_X(o_i) = \text{TRUE}\}$ **do** $W_X \leftarrow W_X + z$
13. $\quad\quad$ **for** each $\{Y \in \mathcal{E} | f_Y(o_j) = \text{TRUE}\}$ **do** $L \leftarrow L \cup Y$
14. $\quad\quad$ **if** $(dist(o_i, o_j) \leq \epsilon)$ **then** $w_\epsilon[X] \leftarrow w_\epsilon[X] + z$
15. $\quad\quad$ **else** $w_\eta[X] \leftarrow w_\eta[X] + z$
16. $\quad\quad j \leftarrow j + 1$
17. $\quad$ **for** each $Y \in L$ such that $w_\epsilon[Y] > 0$
18. $\quad\quad g[X, Y] \leftarrow g[X, Y] + 1$
19. $\quad$ update_pairwise_weights(L, w_ϵ, w_η)
20. **for** each ordered pair $(X, Y) \in \mathcal{E} \times \mathcal{E}$
21. $\quad$ compute significance of $g[X, Y]$ based on W_X and W_X^Y

Function **update_pairwise_weights**(L, w_ϵ, w_η)

1. **for** each $X \in L$
2. $\quad$ **for** each $Y \in L$
3. $\quad\quad$ **if** $w_\epsilon[X] > 0$ **then** $W_Y^X \leftarrow W_Y^X + w_\epsilon[Y] + w_\eta[Y]$
4. $\quad\quad$ **if** $w_\epsilon[Y] > 0$ **then** $W_X^Y \leftarrow W_X^Y + w_\epsilon[X] + w_\eta[X]$

In the worst case the algorithm has to compute the distances between all objects. Thus, the time complexity is $\mathcal{O}(n^2)$, where n is the number of objects. In practice, the number of potential nearby objects of an arbitrary object is a fraction of all objects. Further, the run time is influenced by the proportion of events and unlabelled locations, frequency of them, and bandwidth. We are conducting systematic experiments of the influence of different factors.

Fixing a distance of ϵ is inconvenient (demonstrated in Fig.2). Further, fixing bandwidth h corresponds to the problem that there is usually not a single correct

way of defining H_0 of "no correlation". Thus, it is, for data mining purposes, meaningful to study significance values as function of h (as in Fig.2). For clarity the algorithm was presented above in a form that takes as input only single values for ϵ and h. We extended it to inspect several ϵ_i and h_i in the same run, with a very small increase in computational cost, which is dominated by $\max(h_i)$.

If the data or data structures are too large to be stored in main memory, the hashing strategy introduced by [9] can easily be included as a preprocessing phase. The algorithm can be applied to each bucket separately, and the weights are finally summed up over all of them.

Though the analysis above used the probability $G_{A,B}(\epsilon)$ as the test statistic, the method and algorithm can be extended to the analysis of other test statistics. For instance, the so-called $K_{A,B}(\epsilon)$-function considers the number of events within ϵ, instead of the nearest neighbour only.

When testing the significance of correlation between point patterns of two event types, spatial autocorrelation of event types may be a distorting factor. Though the selected H_0 assumes independence *between* the event types, it should allow dependence of events that are instances of the same event type. In our method, the binomial distribution of $G_{A,B}(\epsilon)|H_0$ was based on the assumption of independence of the samples. However, the distorting effect of possible autocorrelation is reduced by two facts. First, though the samples are independent, the autocorrelation is implicitly taken into account by weighting the samples according to the kernel density estimated intensities. A remarkable concentration of events (i.e., autocorrelation) increases the probability of selecting a sample from the neighbourhood. Second, the testing of rule $A \rightarrow B$ is carried out by fixing the locations of $b \in \mathcal{B}$. Thus, the possible autocorrelation in $\mathcal{B}$ is kept unchanged. We are currently studying the subtle problem of autocorrelation by conducting further experiments with synthetic data.

5 Experiments

5.1 Synthetic Data

We tested the algorithm on synthetic data with 100 event types, each having 1000 uniformly randomly generated events. Different numbers of unlabelled points from zero up to 700,000 were also generated, keeping the locations of the labelled events fixed in each trial. The observation region was a square of size $100,000 \times 100,000$. We studied the values of $g_{i,j}(\epsilon)$ for each pair $(i,j) \in \mathcal{E} \times \mathcal{E}$, and the significances of the deviation from the distribution of $G_{i,j}(\epsilon)|H_0$ the algorithm assigned to them.

Since the data are independently generated, the average deviation from H_0 should be zero. As described in Sec. 3, the estimates are biased with small S. When increasing the number of unlabelled points, the average tended to zero. Fig. 4 summarizes the results in the case that S included the maximum of 700,000 unlabelled points, and the 100,000 labelled events. There is a dot for each ordered pair $(i,j) \in \mathcal{E} \times \mathcal{E}$. The x-axis indicates the value $g_{i,j}(1000)$. The y-axis shows the significance of the deviation from the H_0-distribution that our

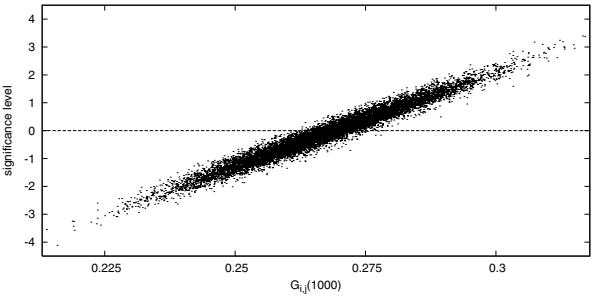


Fig. 4. Trial on synthetic data (100 event types, each with 1000 randomly generated events, 700,000 unlabelled points). Each dot represents an ordered pair (i, j) of event types: $g_{i,j}(1000)$ (x-axes) vs. the significance of deviation of $g_{i,j}(1000)$ from $\hat{G}_{i,j}(1000)|H_0$ of no spatial correlation (y-axes) assigned by the proposed algorithm.

algorithm assigned to each $g_{i,j}(1000)$. The range [-1.96,1.96] corresponds to 95 % confidence interval. The kernel function was the bivariate Gaussian, $h = 5000$ (symmetric, $\rho = 0$). The average significance was very slightly biased (0.08). The running time (Pentium 1.33 GHz Linux) was 2 h 25 min. Another test run with a larger generated data (500 event types, 1000 events in each, resulting in 500,000 labelled events, and $|S| = 1,000,000$) took 5 1/2 hours.

Given a value of the x-coordinate, the range of values of the y-coordinate indicates the range of different assessments of significance, the value of $g_{i,j}$ being the same. This spread is due to the statistical error inherent in any bootstrap approach, caused by the difference of population F, and sample S. Though the intensity of the generating process is constant, random variation causes events of some event types to be located, on average, in regions of larger overall density, resulting in different significance assessments.

5.2 Real Data: Linguistic Analysis of Place Names

The Place Name Register maintained by the National Land Survey of Finland contains 717,746 Finnish place names, including the coordinates and types of the named locations. Each named object is represented by a pair of coordinates.

Linguistic features appearing in place names are of interest for many different research fields. Place names preserve features that have disappeared from the current language. Thus, they are significant for the research on, e.g., the history of languages. Loan words are signs of interaction between cultures, and provide material for the research on cultural history and history of settlement. As an example, Finnish place names of Saami origin give evidence for Saami inhabitants in South and Central Finland during the Iron Age.

In many languages, e.g., in German and Finnish, place names are typically compound words. A natural approach is to consider the different lexemes (words) as different event types, and to study their relationships. As a tedious preprocessing

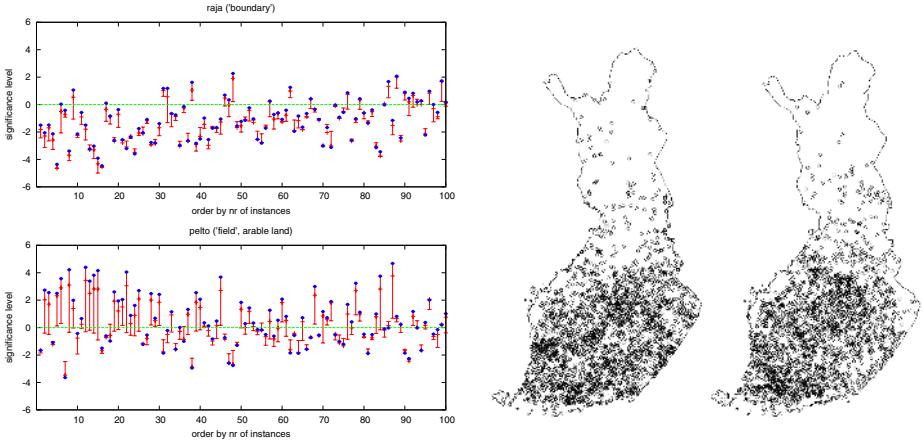


Fig. 5. Events of name elements *raja* 'boundary' (left-side map), and *pelto* 'arable land' (right-side map). Summary of significances of rules $raja \rightarrow B$ (top), $pelto \rightarrow B$ (bottom), $\epsilon = 1.5$ km, where B is in turn each of the 100 most common name elements indicated by x-axis in decreasing order of the number of instances. Errorbars indicate the different bandwidths of Gaussian kernel functions: 1500, 5000, and 10,000 metres, the widest bandwidth indicated by points.

phase of the data, we first extracted the individual name elements from the compound names, and found the basic forms of the inflected words.

Since the endings and first parts of compound names have different semantic functions, it is useful to analyze them separately. In the following we only consider the first parts. We studied the pairwise correlation patterns between the name elements such that the number of instances as the first parts of compound names is at least 30. The number of event types was 1,707. The running time of the algorithm was 37 minutes (1.33 GHz Linux Pentium) when using the Gaussian kernel, $h = 5000$ metres. For a detailed description of the results of the analysis, see [7].

We illustrate the analysis of profiles of correlation patterns by an example. Fig. 5 displays the instances of two name elements: *raja*, meaning 'boundary', and *pelto*, meaning 'field' (arable land). Though both event types have approx. 2,500 instances occurring in the same (larger) areas, they differ remarkably in a local level. Instances of *raja*, unlike those of *pelto*, are not located in the immediate neighbourhood of the other common name elements as indicated by Fig. 5. In this case the meaning of *raja*, boundary, gives reason to the explanation that the instances are located apart from the heart of the (earlier) settlement, unlike those of *pelto*, referring to arable land.

6 Conclusion

The methods of spatial data mining and spatial statistics have been quite separated when it comes to finding associations in spatial point data and evaluating

their significances. While data mining approaches have concentrated on developing algorithms in the spirit of association rules and frequent patterns, the statisticians have been working with point process models leading to tedious simulations for evaluating attraction and repulsion patterns in data.

We develop an intermediate approach, and introduce an algorithm for evaluating associations of features in large spatial data sets with even thousands of features. Compared to collocation rules, the associations found by the developed algorithm provide more detailed knowledge, since they take characteristics of the whole data set into account when assessing the significance of an observed association.

We tested the methods on synthetic data, and a large place name data set. The different name elements appearing in compound words were extracted, and they were treated as features with a location. The number of name elements is large, and thus, novel methods are needed in the analysis.

References

1. A. C. Davison, D.V.Hinkley. *Bootstrap Methods and their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.
2. P. J. Diggle. *Statistical Analysis of Spatial Point Patterns*. Mathematics in Biology. Academic Press, London, 1983.
3. Y. Huang, S. Shekhar, and H. Xiong. Discovering Colocation Patterns from Spatial Data Sets: A General Approach. *IEEE Transactions on Knowledge and Data Engineering*, 16 (12), 1472–1485, December 2004.
4. Y. Huang, H. Xiong, S. Shekhar, and J. Pei. Mining confident co-location rules without a support threshold. In *Proc. 2003 ACM Symposium on Applied computing*, pages 497–501, Melbourne, Florida, 2003.
5. A. Leino, H. Mannila, and R. Pitkänen. Rule discovery and probabilistic modeling for onomastic data. N. Lavrac, D. Gamberger, L. Todorovski, H. Blockeel (eds.), Knowledge Discovery in Databases: PKDD 2003, 291–302. Lecture Notes in Artificial Intelligence 2838. Springer, 2003.
6. M. Salmenkivi. Evaluating attraction in spatial point patterns with an application in the field of cultural history. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 511-514. Brighton, UK, November 2004.
7. M. Salmenkivi, S. Hyvönen, A. Leino, and H. Tuominen. Computational survey of clustering in Finnish place name elements. *Proc. of 22nd International Conference on Onomastic Sciences, ICOS XXII*. Pisa, Italy, August–September 2005.
8. S. Shekhar and Y. Huang. Discovering spatial co-location patterns: a summary of results. In *Proceedings of 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2001)*, Redondo Beach, CA, USA, 2001.
9. X. Zhang, N. Mamoulis, D. Cheung, Y. Shou. Fast mining of spatial collocations. In *Proc.10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 384–393, Seattle, Washington, 2004.

Improving Functional Modularity in Protein-Protein Interactions Graphs Using Hub-Induced Subgraphs*

Duygu Ucar[1], Sitaram Asur[1], Umit Catalyurek[2], and Srinivasan Parthasarathy[1,2]

[1] Department of Computer Science and Engineering, The Ohio State University
[2] Department of Biomedical Informatics, The Ohio State University
srini@cse.ohio-state.edu

Abstract. Dense subgraphs of Protein-Protein Interaction (PPI) graphs are believed to be potential functional modules and play an important role in inferring the functional behavior of proteins. PPI graphs are known to exhibit the scale-free property in which a few nodes (hubs) are highly connected. This scale-free topology of PPI graphs makes it hard to isolate dense subgraphs effectively. In this paper, we propose a novel refinement method based on neighborhoods and the biological importance of hub proteins. We show that this refinement improves the functional modularity of the PPI graph and leads to effective clustering into dense components. A detailed comparison of these dense components with the ones obtained from the original PPI graph reveal three major benefits of the refinement: i) Enhancement of existing functional groupings; ii) Isolation of new functional groupings; and iii) Soft clustering of multi-functional hub proteins to multiple functional groupings.

1 Introduction

Protein-Protein interaction (PPI) graphs have been obtained through accumulations of experimentally determined interactions between proteins. The presence of biologically relevant functional modules in PPI graphs has been theorized by many researchers [8,14]. Mining these graphs to isolate functional modules is a crucial task for the purposes of function prediction and identification in computational proteomics. However, extraction of these functional modules using traditional mining/clustering algorithms has proven to be difficult [21,24].

The primary property of the PPI graph that is detrimental to traditional graph mining is its scale-free topology [22] with the degree distribution following the power law as $F(k) \sim k^{\alpha}$ where $\alpha < 0$. Most proteins in the graph participate in a small number of interactions while a few proteins, known as hubs, are involved in a large number of interactions. The topology typically consists of a giant central core containing a significant amount of proteins and their

* This work is supported in part by the DOE Early Career Principal Investigator Award No. DE-FG02-04ER25611 and NSF CAREER Grant IIS-0347662.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 371–382, 2006.
© Springer-Verlag Berlin Heidelberg 2006

interactions. The rest of the proteins are either completely disconnected or part of small disconnected groups. The scale-free topology, makes isolation of modules hidden inside the central core all but impossible [21].

Another challenge in clustering PPI graphs is the need to assign proteins to different groups (soft clustering) based on their functions. Hub proteins typically have multiple functions and are likely to be essential for the organism. Recently, Karypis *et al* [1] presented several multi-level graph partitioning algorithms to address the difficulty of partitioning scale-free graphs. Although the proposed algorithms result in better groupings compared to traditional algorithms, they still do not perform soft clustering.

In order to address these issues, we suggest a key refinement of the PPI graph, motivated by the topological and biological importance of the hub proteins [15]. Our aim is to target the neighborhood of these potentially multi-faceted proteins and isolate, for each of their functions, corresponding densely connected regions. Our approach consists of two stages. In the first stage, we refine the PPI graph to improve functional modularity, using hub-induced subgraphs. We employ the Edge betweenness measure [19] to identify dense regions within the neighborhoods. In the second stage, we cluster the refined graph using traditional algorithms. Our end goal is to isolate components with high degree of overlap with known functional modules. An additional advantage of the refinement process is its ability to perform soft clustering of hub proteins.

Earlier approaches that focus on elimination of hubs from the scale-free graphs have found that this disconnects the graph and breaks down the modules as well [2,12]. Recently, Costa [11] introduced a hub-centered community detection algorithm. We believe that this will not be effective in scale-free graphs since hubs have a large number of neighbors which cannot all be part of the same community. To the best of our knowledge, we are the first to suggest duplicating hubs to improve modular decomposition of scale-free graphs. Although, in this work, we focus on PPI graphs, our refinement technique is applicable to any scale-free graph.

Other groups have attempted to extract dense regions to isolate protein complexes from PPI graphs [5,17] using concepts such as k-cores or cliques. Although dense regions of the PPI graph are highly associated with known functional modules, they are by themselves not entirely informative in terms of function prediction. Mining the entire PPI graph will definitely prove to be a superior source for novel discovery of protein functions. Hence, we aim to improve the modularity of a PPI graph, as a whole, which enables enhanced functional prediction/identification of every protein of the graph.

The proposed refinement technique is evaluated on the PPI graph of Saccharomyces Cerevisiae obtained from the DIP (Database of Interacting Proteins) database. In order to quantify the quality of our clustering, we employ both topology-based and domain based validation metrics. We find that the clusters we obtain after refinement match very well with known biological annotations. In addition, we obtain groupings after refinement that could not be obtained from the original graph. Our technique also allows soft clustering of multi-functional

proteins. We find that each of these clusters include proteins sharing a certain function with the multi-functional protein.

2 Graph Refinement

2.1 Evolutionary Implications

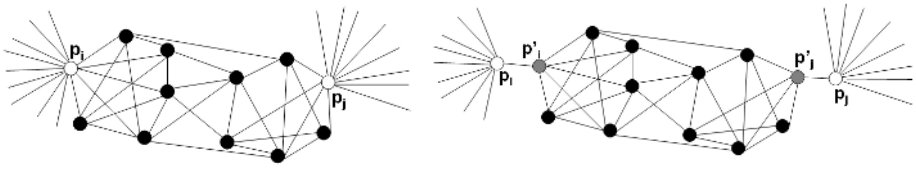
Recently, several groups [6,10,25] have suggested mathematical models to explain the evolutionary growth of Protein-Protein interactions graphs. They claim that preferential attachment is one of the main causes for the scale-free topology of interaction graphs. According to the duplication-divergence model proposed by Vazquez *et al* [25], there is a linear relation between a node's degree and the probability of a new node attaching to that node, known as preferential attachment. Since hubs have very high degrees, new proteins added to the graph are more likely to interact with hubs rather than other nodes. Hence, if a hub belongs to a functional module, most of the other proteins in that module will prefer to connect to the hub rather than a node with the same function but less degree. This suggests that proteins with the same function interact within themselves and also individually with at least one hub. For this reason, we believe that it is important to consider neighborhoods of hubs to isolate functional modules.

2.2 Hub-Induced Subgraphs

As stated before, hubs typically tend to be essential proteins [15], having several important functions inside the cell. Hubs can therefore be linked to several functional modules. However, most of their interactions do not imply a functional similarity. In this work we are aiming to identify the neighbors of hubs that share functionalities with the hub protein. Hence, our goal is to identify all dense components that lie within the neighborhood of a hub. Once such components are identified, the neighboring hub is duplicated and all its interactions with the members of the group reassigned to the duplicate. In addition, all the duplicates will be linked to the original hub to preserve the original interactions of the proteins belonging to the isolated dense component. Note that we are not eliminating any interactions. We are merely re-assigning interactions between the proteins belonging to the dense components and the hub to the duplicate.

If the proteins of a functional module are divided across neighborhoods of several hubs, each of those hubs will be duplicated once and will be included in the functional module. This will isolate the functional module from the unrelated neighbors of the hubs and create a tightly knit group. An example can be seen in Figure 1.

We perform duplication of hubs into several new nodes for each dense component in the hub's neighborhood. In order to identify these dense components, we introduce the notion of a hub-induced subgraph.



(a) Example graph (b) Graph after hub duplications

Fig. 1. Illustration of Hub Duplication. Proteins p_i and p_j are duplicated and the duplicates, p_i and p_j are connected to dense components as well as the original proteins.

Definition 1. Let $G = (V, E)$ be a graph. $G' = (V', E')$ is a *vertex-induced subgraph* of G if $V' \subseteq V$ and E' is all the edges of G between elements of V'.

Definition 2. A *hub-induced subgraph* of G is a graph $G'' = (V'', E'')$, where V'' corresponds to a hub's adjacency list.

Thus, for every hub of the graph, there exists a corresponding hub-induced subgraph obtained from the adjacency list of the hub. We isolate these hub-induced subgraphs to identify potential functional modules. For more details on our analysis of these hub-induced subgraphs, please refer our technical report [23].

2.3 Hub Duplication

To obtain information about the neighborhoods of hubs, we use the Edge betweenness measure which was first introduced by Newman *et al* [19]. This measure favors edges between communities and disfavors ones within communities. Newman *et al* introduced three different Edge betweenness measures; Shortest-path, Random-walk and Current-flow. We use the Shortest-path betweenness measure, which considers the number of shortest paths between all pair of nodes going along each edge.

Given a graph, $G(V, E)$ and 'known number of partitions'(k), the algorithm identifies k sub-groups such that the intra-group connections are dense and inter-group connections sparse, by repetitively removing edges with high Betweenness values. Our goal is to detect dense regions inside each hub-induced subgraph. We implement the algorithm without the k parameter and include the clustering coefficient of the subgraphs as the stopping criteria.

The clustering coefficient [26] is a measure that represents the interconnectivity of a vertex's neighbors. The clustering coefficient of a vertex v is defined as the proportion of edges between its direct neighbors to the number of edges that could possibly exist between them. The clustering coefficient for the whole graph is the mean over the coefficients of all vertices in it and lies between 0 and 1. Tightly knit groups are associated with high clustering coefficients.

Although the Shortest-path betweenness algorithm is computationally costly ($O(E^2V)$ running time), since the hub-induced subgraphs are small in size ($<$ 284 nodes), it is tractable for our purpose. The pseudo-code of our refinement

Algorithm 1. Identify-Dense-Regions(G_i)

```
INPUT G_i = (V_i, E_i) : hub-induced subgraph of Hub_i
if size(G_i) < T_size then
  Return
else if CC(G_i) ≥ T_cc then
  DuplicateHub(Hub_i,G_i)
else
  e = most-between-edge(G_i)
  //remove e from G_i
  G_i ← G_i − e
  recalculate Edge betweenness values
  if G_i is partitioned into G_i^1 and G_i^2 then
    Identify-Dense-Regions(G_i^1)
    Identify-Dense-Regions(G_i^2)
  else
    Identify-Dense-Regions(G_i)
  end if
end if
```

algorithm with Shortest-path betweenness and clustering coefficient measures is given in Algorithm 1. Here, *most-between-edge(G_i)* returns the edge with the highest Shortest-path betweenness score in the G_i subgraph. T_{size} represents the minimum size(number of nodes) of the dense components we will consider and T_{cc} represents the clustering coefficient threshold. When a component (of size $>= T_{size}$) is dense enough, algorithm calls *DuplicateHub* function to duplicate the corresponding hub and re-assign its interactions with the members of the dense component to the duplicate. For each dense component identified from a hub-induced subgraph a duplication event takes place.[1]

3 Methods

3.1 Clustering Algorithms

Once the PPI graph is refined using hub-induced subgraphs, the resulting graph is clustered to separate out the functional modules. We used two graph clustering algorithms - a single-level Spectral algorithm and kMETIS [16], a multi-level partitioning algorithm. The Spectral-based algorithm uses Eigenvectors of the Laplacian matrix constructed from the graph to determine effective clusters of the graph. These clusters minimize the total weight of the edge cut. The kMETIS algorithm, obtains a k-way partition of the approximate graph and refines it to construct a k-way partitioning of the original graph. For more details about these algorithms please refer our technical report [23].

[1] Note that the Betweenness scores are recalculated whenever an edge is removed from the graph to capture the topology of the remaining graph.

3.2 Validation Measures

Topological Measure. To evaluate our clusters, we use a topology-based modularity metric proposed by Newman [19]. This metric considers a k X k symmetric matrix of clusters where each element a_{ij} represents the fraction of edges that link nodes between clusters i and j and each a_{ii} represents the fraction of edges linking vertices within cluster i. The modularity measure is given by

$$M = \sum_i (a_{ii} - (\sum_j a_{ij})^2) \tag{1}$$

Statistical Measure based on Domain Information. To test if the clusters obtained correspond to known functional modules, we need to validate our dense components using known biological associations. We used the Gene Ontology Consortium Online Database [4] to look for biological relations between proteins assigned to the same cluster. The Gene Ontology (GO) is a controlled vocabulary designed to accumulate the result of all investigations in the area of genomics and biomedicine by providing a large database of known associations containing common terminology that can be used among researchers. GO provides three ontologies - cellular component(CC), molecular function(MF) and biological process(BP). Cellular component terms refer to the localization of proteins inside the cell. Molecular function terms refer to shared activities at the molecular level and biological process terms refer to entities at both the cellular and organism levels of granularity. We used all three annotations for validation and comparison in accordance with earlier works [24,3].

Merely counting the proteins that share an annotation will be misleading since the underlying distribution of genes among different annotations is not uniform. Hence, we use p-values to calculate the statistical significance of a group of proteins that share a GO term. The p-values essentially represent the chance of observing that particular grouping, or better, given the background distribution. Assume we have a cluster of size n, out of which m proteins share a particular annotation. Also, assume there are N proteins in the database with M of them known to have that same annotation. Then using the Hypergeometric Distribution, the probability of observing m or more proteins that are annotated with the same GO term out of n proteins is:

$$p - value = \sum_{i=m}^{n} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}} \tag{2}$$

Smaller p-values imply that the grouping is not random and is more significant biologically than one with a higher p-value. A cut-off parameter (alpha level) is used to differentiate significant groups from the insignificant ones. If a group of proteins are associated with a p-value greater than the cut-off, they are considered insignificant. We used the recommended cut-off of 0.05 for all our validations.

As the p-value of a single cluster is statistically not representative, we define a Clustering score function in order to quantify the overall clusters. We defined this score as follows.

$$Clustering\ score = \frac{\sum_{i=1}^{n_S} min(p_i) + (n_I * cutoff)}{n_S + n_I} \quad (3)$$

where n_S and n_I denotes the number of significant and insignificant clusters, respectively. $cutoff$ stands for the alpha level(0.05) whereas $min(p_i)$ denotes the smallest p-value of the significant cluster i. Hence, each cluster is associated with one p-value for each of the three ontologies.

4 Experimental Results

In this section, we discuss our experimental results.

Topology-based Evaluation: First, we use the Modularity metric on the clusters obtained using both the kMETIS and Spectral algorithms. Figure 2-b shows

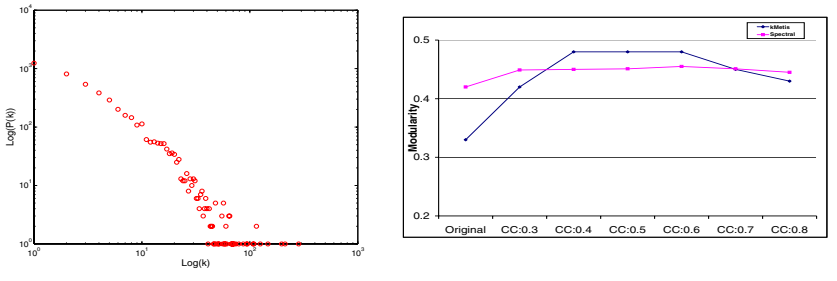
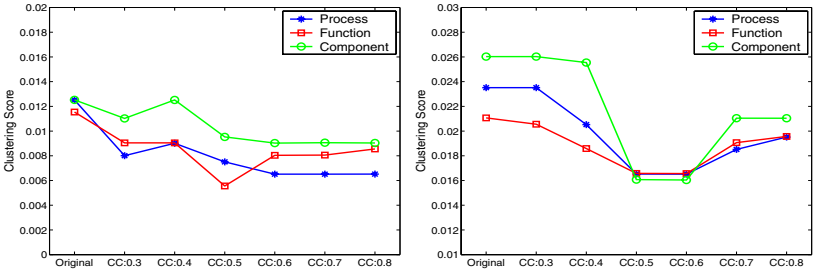


Fig. 2. a) Degree distribution of DIP dataset b) Modularity scores before(*Original*) and after ($CC : 0.3$ to $CC : 0.8$) refinement

the modularity comparison between the original graph and refined graphs for the two algorithms. We find that the refinement improves the modularity of the graph for both algorithms. From the curve, we find that the modularity scores peak at clustering coefficient values between 0.4 and 0.6 in both cases. Further, the modularity for clustering the original graph is much lower than for any of the refined graphs. kMETIS produces clusters with higher modularity(upto 45% better than the original) than Spectral(upto 8% better than the original) for the refined graphs.

Domain-based Evaluation: Next, we test the effectiveness of our refinement technique by comparing with the clusters obtained from the original graph. The DIP dataset consists of 15147 interactions among 4741 proteins. In our work, we analyzed the degree distribution of the graph (shown in Figure 2-a) and defined all nodes of degree greater than 25 (2% of all nodes) to be hubs. We ran the algorithm to find all dense components within every hub-induced subgraph using the clustering coefficient and size as stopping criteria. We chose 6 as the size threshold for dense components, since components with size smaller than 6 are likely to be insignificant. To choose a suitable threshold for the clustering

coefficient, there are two things that we should consider. First, we want the resulting components to be dense enough to correspond to a functional module. We vary the clustering coefficient parameter (T_{cc}) between 0.3 and 0.8 and obtain refined graphs for each. We believe that, considering the incompleteness in PPI graphs and the need for obtaining dense components, a reasonable clustering coefficient value would be around 0.5-0.6. Components that have clustering coefficients within this range are likely to be dense enough to be considered as functional groups and would not be affected too much by the incomplete nature of the dataset. The refined graphs and the original graph are clustered by kMETIS and Spectral clustering algorithms separately. The results obtained are depicted in Figure3. As can be seen from this figure, Clustering scores are reducing (improving) after refinement for both algorithms. Our above hypothesis is validated by the fact that, although an improvement is observed for every clustering coefficient threshold, this improvement is small for low and high values. Also, both algorithms have their smallest Clustering scores for the threshold values of 0.5, 0.6 and 0.7. If we consider the improvement in this clustering coefficient range, our refinement technique improves Clustering scores up to 52%, 48% and 28% for MF, BP and CC ontologies in the case of kMETIS and 30%, 21% and 38% for the same three ontologies for the Spectral algorithm. This confirms that kMETIS produces better clusters than the Spectral algorithm. Note that our Clustering score considers both significant and insignificant clusters. Next,

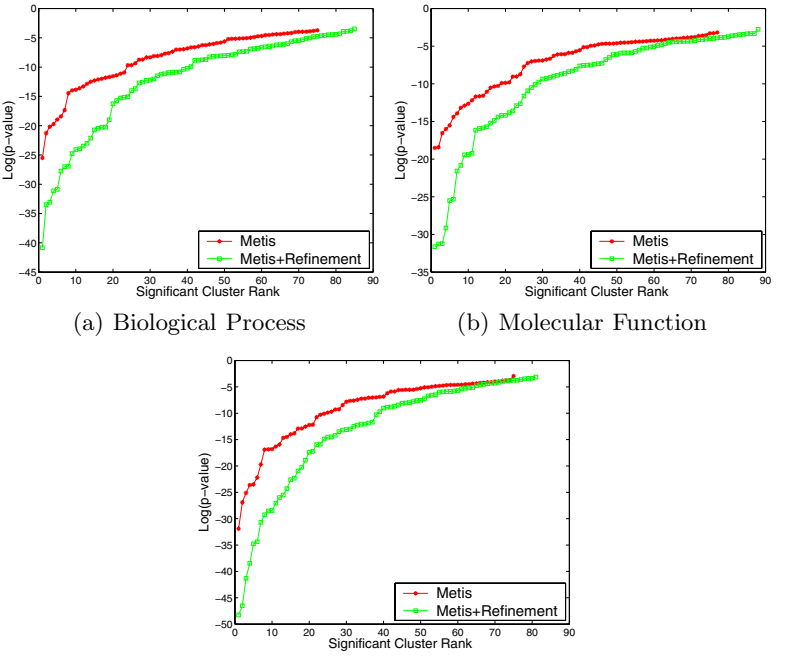


(a) Metis (b) Spectral

Fig. 3. Clustering scores before and after refinement algorithm using kMETIS and Spectral clustering algorithms. *Original* refers to the PPI dataset before the refinement. $CC : 0.3$, $CC : 0.4$,$CC : 0.5$,$CC : 0.6$, $CC : 0.7$, $CC : 0.8$ and $CC : 0.9$ refer to refined graphs with the respective clustering coefficient threshold. Process, Function and Component represent Biological Process, Molecular Function and Cellular Component ontologies respectively.

we evaluate the significance of our clustering results for all three ontologies. In Figure 4(a-c) we show the p-value distribution of significant clusters in both original and refined graphs for all three ontologies. For all ontologies, we find that the refined graph can be clustered into more biologically meaningful groups. For example, the best cluster we obtained on the original graph had a p-value of 8.2089e-25 for Biological Process, whereas the best cluster after refinement

had a p-value of 5.4658e-41 for the same ontology. We obtained similar results for the other two ontologies. In addition, we are able to identify more significant clusters after the refinement for all three ontologies.



(a) Biological Process (b) Molecular Function

(c) Cellular Component

Fig. 4. P-value distribution of significant clusters before and after the refinement. The y axis represents the log(p-value) for each corresponding cluster.

5 Discussion and Conclusion

In this paper, we have proposed a refinement technique to improve modular decomposition of PPI graphs. We refined the PPI graph based on Shortest-path betweenness and clustering coefficient measures. From our experimental results, we found that duplicating the hubs of a scale-free PPI graph improves the modularity of the graph. Thus, we are able to obtain topologically and biologically more significant clusters even using traditional clustering algorithms. A detailed examination of the obtained clusters revealed that the proposed method has three major benefits:

- Enhancement of available functional groupings: We obtain larger groups of proteins that are annotated with the same GO term from our refined graph than on the original graph.
- Isolation of new functional groupings: We find groupings of proteins that could not be obtained from the original graph.

– Soft-clustering: Our approach can identify multi-functional hub proteins and group them into modules corresponding to each of their functions.

We now provide some illustrations from our results for each of these cases. For more details, please refer our technical report [23]: KAP95 (karyopherin beta), an essential protein is known to take part in 'nucleocytoplasmic transport'. Specifically, it participates in a complex mediating nuclear import via a localization signal(NLS). It interacts with nucleoporins to guide transport across the nuclear pore complex [13]. When we cluster the original DIP dataset, this protein is correctly grouped with 8 proteins that are also annotated with 'nucleocytoplasmic transport' term with p-value 1.14e-08.

Using our refinement technique, KAP95 is duplicated once. The hub and its duplicate appear in two separate clusters when we use the kMETIS algorithm. In one, KAP95 is grouped with proteins 18 other proteins that share the same biological process ('nucleocytoplasmic transport') with p-value 1.07e-27. The major difference between this group and the one from the original graph are the inclusion of NUPs(Nucleoporins - 8 proteins) and KAPs(Karyopherins - 3 proteins). Transport through the nuclear pore complex is facilitated by transient interactions between the KAPs and the nuclear pore complex proteins (NUPs) [20]. Thus, locating NUPs and KAPs together is a noticeable benefit caused by our refinement. Clearly, our approach groups more proteins that belong to the same functional module together. This suggests that hub duplications make isolation of modules easier. These clusters are also valuable for predicting the functions of unknown proteins. In the above group, four proteins (YKL061W,YKR064W, YNL122C, YER004W) do not have a known function. Among these four, YKL061W is predicted by Brun et al [7] to take part in 'nucleus-cytoplasm transport' process which is in accordance with our findings. Since two different datasets and approaches are used to infer the same conclusion about protein YKL061W, the overlap is noteworthy. This also suggests that the other three proteins might have an unrevealed task in 'nucleocytoplasmic transport' biological process.

In addition to enhancing clusters, our method is able to assign hub proteins which were originally in insignificant clusters into significant clusters. To illustrate this, we consider the hub protein LSM8. The LSM(Sm-like) proteins interact with each other and with U6 snRNA complex and influence pre-mRNA splicing [18]. In the original dataset, this protein is assigned to a cluster which does not have any significant annotations. However, after the refinement, this protein is located into a cluster which has a biological process annotation with p-value 1.2e-12. In addition to LSM8, ten other proteins in this group are associated with 'mRNA splicing'. LSM8 is located with the members of its complex (other SM-like proteins) as well as the components of U6 snRNP complex(PRP proteins). This example shows that our technique not only improves functional modules which can be identified from the original dataset, but also allows detection of functional modules which cannot be discovered from the original dataset.

Another advantage of our refinement technique is its ability to perform soft clustering on certain hub proteins. CKA1 is one of these multi-faceted proteins

and is involved in several cellular events such as maintenance of cell morphology and polarity, and regulating the actin and tubulin cytoskeletons [9]. When the original dataset was clustered, CKA1 and seven other proteins, annotated with 'transcription, DNA-dependent' term are located in the same cluster(p-value 3.47e-05). On the other hand, our algorithm duplicates CKA1 twice. When we cluster, these 3 nodes are then assigned to different clusters resulting in three different groupings for protein CKA1. All three correspond to different functional modules of the CKA1 protein. One of these clusters is an enhancement of the 'transcription, DNA-dependent' functional module(very low p-value of 2.3e-19). The second cluster includes proteins which are annotated with the biological process term 'protein amino acid phosphorylation' with p-value 1.2e-05. CKA1 is itself annotated with the same term. The third cluster contains 21 proteins and CKA1, all of which are annotated for 'organelle organization and biogenesis' (with p-value 3.2e-12). Thus, we found that our technique, not only improved the obtainable clusters (by decreasing p-value from 3.47e-05 to 2.3e-19), but also grouped CKA1 with proteins that share its different functions. Altogether these examples indicate the effectiveness of our approach on isolation of functional modules from the PPI graphs.

Although, in this work, we have applied our refinement technique to PPI graphs, we strongly believe that it will be equally effective for all other scale-free graphs such as social networks.

References

1. A Abou-Rjeili and G Karypis. Multilevel algorithms for partitioning power-law graphs. *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.
2. R Albert, H Jeong, and A L Barabasi. Error and attack tolerance in complex networks. *Nature*, 406:378–382, 2000.
3. V Arnau, S Mars, and I Marin. Iterative cluster analysis of protein interaction data. *Bioinformatics*, 21:3:364–378, 2005.
4. M Ashburner and *et al.* Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet.*, 25(1):25–29, May 2000.
5. G D Bader and C WV Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinfomatics*, 4:2, 2003.
6. J Berg, M Lssig, and Andreas Wagner. Structure and evolution of protein interaction networks: a statistical model for link dynamics and gene duplications. *BMC Evolutionary Biology*, 4:51, 2004.
7. C Brun, F Chevenet, D Martin, J Wojcik, A Gunoche, and B Jacq. Functional classification of proteins for the prediction of cellular function from a protein-protein interaction network. *Genome Biology*, 5, 2003.
8. C Brun, C Herrmann, and A Guenoche. Clustering proteins from interaction networks for the prediction of cellular functions. *BMC Bioinformatics*, 5(95), July 2004.
9. D A Canton and D W Litchfield. The shape of things to come: An emerging role for protein kinase ck2 in the regulation of cell morphology and the cytoskeleton. *Cell Signalling*, 18:267–275, 2006.

10. F Chung, L Lu, T G Dewey, and D J Galas. Duplication models for biological networks, 2002.
11. L F Costa. Hub-based community finding. *http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cond-mat/0405022*, 2004.
12. P Crucitti, V Latora, M Marchiori, and A Rapisarda. Error and attack tolerance of complex networks. *Physica A*, 340:388–394, 2004.
13. D Gilchrist and M Rexach. Molecular basis for the rapid dissociation of nuclear localization signals from karyopherin alpha in the nucleoplasm. *J. Biol. Chem*, 278:51:51937–51949, 2003.
14. J Hua, D Koes, and Z Kou. Finding motifs in protein-protein interaction networks. *Project Final Report, CMU www.cs.cmu.edu/ dkoes/research/prot-prot.pdf*, 2003.
15. H Jeong, S P Mason, A L Barabasi, and Z N Oltvai. Lethality and centrality in protein networks. *Nature. 411:44.*, 411:41–42, 2001.
16. G Karypis and V Kumar. Unstructured graph partitioning and sparse matrix ordering system. technical report. *http://www-users.cs.umn.edu/ karypis/metis/metis/files/manual.pdf*.
17. X-L Li, S-H Tan, C-S Foo, and S-K Ng. Interaction graph mining for protein complexes using local clique merging. *Genome Informatics*, 16(2):260–269, 2005.
18. A E Mayes, L Verdone, P Legrain, and J D Beggs. Characterization of sm-like proteins in yeast and their association with u6 snrna. *EMBO J.*, 18(15):4321–4331, 1999.
19. M E J Newman and M Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.
20. L F Pemberton and B M Paschal. Mechanisms of receptor-mediated nuclear import and nuclear export. *Traffic*, 6:187, 2005.
21. A L Barabsi S Yook, Z N Oltvai. Functional and topological characterization of protein interaction networks. *Proteomics*, 4:928–942, 2004.
22. A Thomas, R Cannings, N A M Monk, and C Cannings. On the structure of protein-protein interaction networks. *Biochemical Society Transactions*, 31:1491–1496, 2003.
23. D Ucar, S Asur, U Catalyurek, and S Parthasarathy. Improving functional modularity in protein-protein interactions graphs using hub-induced subgraphs. *http://www.cse.ohio-state.edu/research/techReport.shtml. Electronic report under 2006/TR41.pdf*, 2006.
24. D Ucar, S Parthasarathy, S Asur, and C Wang. Effective preprocessing strategies for functional clustering of a protein-protein interactions network. *IEEE, International Symposium on Bioinformatics and Bioengineering, BIBE*, 2005.
25. A Vazquez, A Flammini, A Maritan, and A Vespignani. Modeling of protein interaction networks. *Complexus*, 1:38, 2003.
26. D Watts and S Strogatz. Collective dynamics of small world networks. *Nature*, 393(6684):440–442, June 1998.

Refining Aggregate Conditions in Relational Learning

Celine Vens, Jan Ramon, and Hendrik Blockeel

Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, 3001 Leuven, Belgium
{celine.vens, jan.ramon, hendrik.blockeel}@cs.kuleuven.be

Abstract. In relational learning, predictions for an individual are based not only on its own properties but also on the properties of a set of related individuals. Many systems use aggregates to summarize this set. Features thus introduced compare the result of an aggregate function to a threshold. We consider the case where the set to be aggregated is generated by a complex query and present a framework for refining such complex aggregate conditions along three dimensions: the aggregate function, the query used to generate the set, and the threshold value. The proposed aggregate refinement operator allows a more efficient search through the hypothesis space and thus can be beneficial for many relational learners that use aggregates. As an example application, we have implemented the refinement operator in a relational decision tree induction system. Experimental results show a significant efficiency gain in comparison with the use of a less advanced refinement operator.

1 Introduction

In relational learning, predictions for an individual are based not only on its own properties but also on the properties of a set of related individuals. Many systems use aggregates to summarize this set. Features are then constructed by comparing the result of the aggregate function to a threshold, we call such a feature an aggregate condition. For example, in the context of a data set on parents and children, a possible feature could be *"the maximum age of the person's children is larger than 10"*.

Many learning systems rely on a general-to-specific ordering of the hypotheses to traverse the hypothesis space in an efficient way. Only few of the existing systems that learn hypotheses with aggregates extrapolate this generality ordering to the aggregate conditions, and when they do, they do it in a restricted way. For instance, the feature just mentioned could be refined in three ways: by changing the aggregate function (e.g., change *maximum* into *average*), the subset aggregated over (e.g., specialize *children* into *daughters*) or the threshold compared with (e.g., increase *10* to *15*). No current relational learners consider all three kinds of refinements, and indeed the effect of such refinements on the generality of a rule, and the interaction between these effects, are non-trivial and have not been studied up till now.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 383–394, 2006.
© Springer-Verlag Berlin Heidelberg 2006

This paper presents the first comprehensive study of these effects and interactions. This study leads to the description of a refinement operator that enables relational learners to traverse the hypothesis space more efficiently.

The paper is organized as follows. We start by describing several approaches to learning that use aggregates in Sect. 2. Then we elaborate on the monotonicity properties of aggregate conditions and present a general framework for specializing them (Sect. 3). In Sect. 4, we apply the framework to a relational decision tree learner and assess the efficiency gain that this results in. Finally, conclusions and ideas for further research are presented in Sect. 5.

2 Related Work

Perhaps the most closely related work, at first sight, is the research started by Ng et al. [1] on finding frequent itemsets that fulfill constraints with aggregations (for instance, in addition to minimal support, one can impose that the average price of the items in an itemset must be above some threshold). That research also involves studies of the monotonicity properties of aggregates, but the goals are different from ours: the aggregates occur in the constraints defining the hypothesis space rather than in the learned hypotheses; they summarize information over attributes of one entity rather than over different entities; etc. As a result also the methods developed are very different.

On the other hand, there is work that shares our goal of learning hypotheses with aggregates. Here we can distinguish methods that use a fixed set of aggregates defined in advance [2,3,4,5], and methods that construct the aggregates as part of the learning process. The latter, on which we will focus, are especially useful when one wants to consider complex aggregates, where the set to be aggregated over is generated by a complex query: there may be too many such aggregates to compute and store them all during preprocessing. Perlich and Provost [6] provide a detailed examination of aggregation in relational learning and demonstrate that such complex aggregate conditions can significantly improve generalization performance.

Unfortunately, finding good hypotheses with complex aggregate conditions is difficult. Besides the fact that the hypothesis space is significantly expanded by allowing complex aggregate conditions, it also becomes more difficult to search this space in a structured way, because the effect of refinements of the aggregate condition on the generality of the hypothesis is not well-understood in general. As a result, all current relational learners are somehow limited with respect to the aggregates they can learn. Krogel and Wrobel [7] introduce in their propositionalized table aggregate functions that apply not only to single attributes, but also to pairs of attributes, one of which has to be nominal and serves as a *group by* condition. The resulting aggregate conditions are still of limited complexity and are not refined further during the search. Knobbe et al. [8] propose a method for subsequently specializing the set to be aggregated. By restricting the application of this specialization operator to aggregate functions where its effect is well-understood, they can search the hypothesis space in a general-to-specific way, but this obvi-

ously limits the kind of complex conditions that can be found. Van Assche et al. [9] discuss refinement of aggregate conditions in the context of relational decision tree induction. They handle the expansion of the feature space by upgrading the tree learner into a random forest induction system, where random feature subsampling reduces the number of features tested at each node in a tree. Uwents and Blockeel [10] describe relational neural networks as a subsymbolic approach towards learning complex aggregates. Their approach is not constrained to using predefined aggregate functions and does not make a distinction between searching for aggregate functions and searching for complex conditions, but the resulting theories are also not interpretable in terms of well-understood aggregates and conditions.

Clearly none of the existing approaches fully solve the problem of searching general-to-specific in a hypothesis space that may include aggregates of arbitrary complexity. This paper presents the first solution to it.

3 Specializing Aggregate Conditions

ILP systems (inductive logic programming [11]) learn sets of logic clauses, typically by learning one clause at a time. A single clause has the form $h \leftarrow b_1, b_2, \ldots, b_n$, where h and b_i are literals. The ILP system usually finds a clause by starting with the empty clause $h \leftarrow$ and gradually refining it, adding literals to the body of the clause, until it is consistent with the data. It is known that adding literals to the body cannot increase the coverage of the clause, and this can be used to prune the search space. We call refinements that never increase the coverage of a clause, valid refinements.

Now assume that one of the b_i is an aggregate condition of the form $F(\{V|Q\})$ θ R, where F is an aggregate function, Q a conjunctive query, V a variable occurring in Q, θ one of $\{\leq, \geq\}$ and R a numeric value. For instance, the clause

$$person(P, pos) \leftarrow max(\{A|child(P, C), age(C, A)\}) \geq 10$$

classifies a person as positive if the maximum age of his children is higher than 10. Such a clause could now be refined not only by extending the clause itself with a literal b_{n+1}, but also by extending the query Q, or by changing F or R. The question is then under what conditions such refinements are valid. For instance, changing the preceding aggregate condition into

$$max(\{A|child(P, C), age(C, A), male(C)\}) \geq 10$$

is a valid refinement, but with *min* instead of *max* this would not be the case. Also with $\leq$ instead of $\geq$ the above refinement would not be valid.

In this section, we will provide an answer to which refinements of aggregate conditions are valid. We start by discussing aggregate conditions in more detail (Sect. 3.1). Then we present several classes of aggregate functions with an order relation (Sect. 3.2). Afterwards, we elaborate on the refinement of aggregate conditions (Sect. 3.3). Finally, we discuss so-called refinement cubes that visualize possible refinements (Sect. 3.4). Although in this section we have chosen the ILP formalism for expressing relational concepts, the theory behind it can be applied to other relational representations as well.

3.1 Aggregate Conditions

The aggregate conditions that we consider have the following general form: $F(S) \in I$, with F an aggregate function, I a numerical interval, and S a set to be aggregated. For now we will make abstraction of the fact that S is generated by some query Q. More formally, an aggregate condition c takes the following signature: $c : \mathbb{F} \times \mathbb{S} \times \mathbb{I} \rightarrow \mathbb{B}$, with $\mathbb{F}$ a set of aggregate functions (e.g., $\mathbb{F} = \{$count,max,min,sum,avg$\}$), $\mathbb{S}$ a set of sets, $\mathbb{I}$ a set of intervals, and $\mathbb{B}$ the set of boolean values, that indicate whether the condition $F(S) \in I$ holds.

In order to define valid refinements of aggregate conditions we need the concept of *monotonicity*.

Definition 1. *A function* $f(x_1, ..., x_n)$ *is*

- monotone *in* x_i *iff* $x_i < x_{i'} \Rightarrow f(x_1, ..., x_i, ..., x_n) < f(x_1, ..., x_{i'}, ..., x_n)$,
- anti-monotone *in* x_i *iff* $x_i < x_{i'} \Rightarrow f(x_1, ..., x_i, ..., x_n) > f(x_1, ..., x_{i'}, ..., x_n)$,
- *and* non-monotone *in* x_i *otherwise.*

To investigate the monotonicity properties of an aggregate condition $F(S) \in I$, an order relation is needed on the domains $\mathbb{F}$, $\mathbb{S}$, $\mathbb{I}$, and $\mathbb{B}$. We define these relations as follows:

- $\mathbb{F}$: $F_1 \preceq_{\mathbb{F}} F_2 \Leftrightarrow \forall S \in \mathbb{S} : F_1(S) \leq F_2(S)$, this is discussed in the next section,
- $\mathbb{S}$: $S_1 \preceq_{\mathbb{S}} S_2 \Leftrightarrow S_1 \subseteq S_2$,
- $\mathbb{I}$: $I_1 \preceq_{\mathbb{I}} I_2 \Leftrightarrow I_1 \subseteq I_2$,
- $\mathbb{B}$: *false* $\preceq_{\mathbb{B}}$ *true.*

3.2 Ordering the Aggregate Functions

In this section we discuss several parameterized classes of aggregate functions that are ordered and together cover all aggregates of interest. For this paper, we set the aggregate functions of interest to be those defined in standard SQL, i.e., *max*, *min*, *avg*, *sum*, and *count*.

Generalized Averages. We define a class of generalized averages as follows:

Definition 2

$$avg_k(S) = \left(\frac{\sum_i (x_i^k)}{n}\right)^{1/k} \text{ with } S = \{x_1, ..., x_n\}$$

The function $avg_k(S)$ is defined for $-\infty \leq k \leq \infty$ $(k \neq 0)$ if $S \subseteq \mathbb{R}^+$ and for $k = \{1, -1, \infty, -\infty, 2 * z\}$ with $z \in \mathbb{Z}$ if $S \subseteq \mathbb{R}$. If $S \subseteq \mathbb{R}^+$, then the following order relation holds: $i \leq j \Rightarrow avg_i \preceq_{\mathbb{F}} avg_j$. While this relation holds for all k, in practice only some of these k-values are commonly used: $avg_1(S) = avg(S)$, $\lim_{k \to \infty} avg_k(S) = max(S)$, and $\lim_{k \to -\infty} avg_k(S) = min(S)$. Moreover, the order relation $min \preceq_{\mathbb{F}} avg \preceq_{\mathbb{F}} max$ also holds for sets S that contain negative numbers.

Generalized Sums. For *sum* we can define an aggregate function class very similar to the generalized averages:

Definition 3

$$sum_k(S) = (\sum_i (x_i^k))^{1/k} \text{ with } 1 \leq k \leq \infty \text{ and } S = \{x_1, ..., x_n\}$$

We have that $sum_1(S) = sum(S)$ and $\lim_{k\to\infty} sum_k(S) = max(S)$. In other words, these generalized sums range from *max* to *sum*. If S contains only positive numbers, we obtain the following order relation: $i \geq j \Rightarrow sum_i \preceq_{\mathbb{F}} sum_j$.

Generalized Counts. Our last aggregate function of interest is *count*. An aggregate function that can form an aggregate class with *count* is *count distinct*. This function counts the number of *different* values in the set. We have the following order relation: $count\_distinct \preceq_{\mathbb{F}} count$.

Summary. While the proposed aggregate classes contain an infinite amount of aggregate functions, for the most important ones, we obtain the following order: $min \preceq_{\mathbb{F}} avg \preceq_{\mathbb{F}} max$, $max \preceq_{\mathbb{F}} sum$ if $S \subseteq \mathbb{R}^+$, and $count\_distinct \preceq_{\mathbb{F}} count$.

Remark that other parameterized classes for these aggregate functions exist.

3.3 Refinement of Aggregate Conditions

Having defined an ordering relation on the domains $\mathbb{F}$, $\mathbb{S}$, $\mathbb{I}$, and $\mathbb{B}$ of an aggregate condition, we can discuss the monotonicity properties of each domain and define valid refinements.

Monotonicity Properties. For the ease of explanation, we look at an aggregate condition $F(S) \in I$ as the composition of two functions:

- an aggregate function $a : \mathbb{F} \times \mathbb{S} \to \mathbb{R}$,
- a member function $m : \mathbb{R} \times \mathbb{I} \to \mathbb{B}$.

For each of the functions, we can now describe the monotonicity properties. Afterwards, we describe some issues that come along when composing them.

Monotonicity of the aggregate function. The function $a(F, S)$ is monotone in F, because the order on $\mathbb{F}$ is defined as such.

The monotonicity of $a(F, S)$ w.r.t. S depends on F. The function $a(F, S)$ is monotone in S if $F \in \{count, count\_distinct, max\}$ (if any of these functions is applied to a subset $S' \subseteq S$, its resulting value will decrease). Similarly, $a(min, S)$ is anti-monotone, and $a(sum, S)$ and $a(avg, S)$ are non-monotone in S[1].

Monotonicity of the member function. The member function $m(R, I)$ is monotone in I: decreasing the interval can cease the membership of R.

The monotonicity in R depends on I: $m(R, [v, \infty[)$ (with $v \in \mathbb{R}$) is monotone in R, $m(R,]\text{-}\infty, v])$ anti-monotone, and $m(R, [v, w])$ $(v, w \in \mathbb{R})$ non-monotone.

[1] For *sum* the monotonicity depends on the set S, e.g., if this set contains only positive numbers then $a(sum, S)$ behaves monotone.

Monotonicity of the composite function. Care is needed when composing the aggregate function and the member function. The monotonicity properties in F and S are inherited by $m(a(F,S),I)$ if this function is monotone in $a(F,S)$. However, the monotonicity in F and S is reversed if the composed function is anti-monotone in $a(F,S)$. For example, a condition $max(S) \in]\text{-}\infty, v]$ is anti-monotone in max and S.

Similarly, when the composed function is non-monotone in $a(F,S)$, the monotonicity properties in F and S are broken. For instance, for a condition $max(S) \in [v,w]$, the monotonicity properties in max and S are lost. Therefore, in the following we do not consider intervals of the form $[v,w]$, with $v,w \in \mathbb{R}$. We only consider the aggregate conditions $F(S) \leq v$ and $F(S) \geq v$.

Refinement. In order to define valid refinements for an aggregate condition, we can use the monotonicity properties described above. Keeping in mind that the goal is to obtain more specific conditions (i.e. refinements that make the condition false for some of the examples for which it was true), Definition 1 learns that a function that is monotone (anti-monotone) in one of its inputs can be validly refined by decreasing (increasing) its value for that input.

Before turning to an example, we explain how a set S can be increased or decreased in an ILP system.

Increasing or decreasing S. An aggregate condition like $max(S) \geq v$ can be validly refined by decreasing S, i.e., reducing the set to be aggregated. In ILP, this can be achieved by specializing the query Q used to generate the set.

An aggregate condition like $min(S) \geq v$ can be validly refined by increasing S. This can be obtained in ILP by generalizing the query that is used to generate the set, thus, by removing literals from it.

Example. We now illustrate the possible refinements with a small example. Suppose we have the following clause:

$$person(P, pos) \leftarrow max(\{A | child(P,C), age(C,A)\}) \geq 10.$$

The aggregate condition $F(S) \in I$ in this clause can be refined in three ways:

- decrease I (because the member function $m(R,I)$ is monotone in I)

  $$max(\{A | (child(P,C), age(C,A)\}) \geq 15.$$

- decrease $max(S)$ (because for $I = [v, \infty[$ the member function $m(R,I)$ is monotone in R). This can be achieved by
  - decreasing max (since the aggregate function $a(F,S)$ is monotone in F)

    $$avg(\{A | child(P,C), age(C,A)\}) \geq 10.$$
  - decreasing S (since $a(max,S)$ is monotone in S).

    $$max(\{A | child(P,C), age(C,A), male(C)\}) \geq 10.$$

3.4 The Refinement Cubes

We have presented three dimensions along which aggregate conditions can be refined: the aggregate function F, the query Q (or equivalently, the set S_Q, i.e., the set generated by Q), and the interval bound v. The whole set of hypotheses spanned by these three dimensions can be visualized in what we call a *refinement cube* (see Fig. 1). Every discrete point in the cube represents a hypothesis and can be constructed in a finite number of steps, starting from one aggregate condition. A chain of refinements in the cube will be called a *path*. For simplicity, we only consider refinements along one direction at a time. We are only interested in valid refinements, therefore we only allow *monotone paths* in the refinement cubes, i.e. paths that consist only of valid refinements.

For a given aggregate function class, refinement of aggregate conditions proceeds as follows. For every numeric attribute A, we look for a query Q that generates the set of values S_Q for each example. We take the smallest and largest aggregate function in the class (F_{small} and F_{large} respectively) and look for the smallest possible value returned by $F_{small}(S_Q)$ and the largest possible value returned by $F_{large}(S_Q)$ (V_{small} and V_{large} respectively). Then we construct two *start conditions*: $F_{large}(S_Q) \geq V_{small}$ and $F_{small}(S_Q) \leq V_{large}$. For each aggregate function class and each start condition there is a corresponding refinement cube that shows the allowed refinements.

The Refinement Cubes for the Generalized Averages. The generalized averages range from *min* to *max*. Possible thresholds for these functions range from the minimum to the maximum value in the dataset for the attribute under consideration. Hence, the two start conditions for this aggregate class are $max(S_Q) \geq min\_value$ and $min(S_Q) \leq max\_value$. The corresponding refinement cubes are shown in Fig. 1(a), with the start conditions indicated as a large dot. The arrows show the directions in which we can generate monotone paths starting from these aggregate conditions. Observe that when moving along the F-axis, the monotonicity properties of the aggregate function in S_Q change, so moving along the Q axis is only allowed in the top or bottom faces of the cube.

The Refinement Cubes for the Generalized Sums. Figure 1(b) shows the refinement cubes for the generalized sums. The V-axis ranges from the lowest value in the range of *max* (the minimal value for the numeric attribute) to the largest value in the range of *sum* (the maximum of the sum of the values for the attribute A, grouped by example, this value is called sum_value in the cube). The start conditions from which we can generate the whole cube are thus $sum(S_Q) \geq min\_value$ and $max(S_Q) \leq sum\_value$. The second start condition is anti-monotone in S_Q, and therefore positioned at the *specific* side of Q.

In this case moving along the F-axis does not change monotonicity (under the assumption that the generalized sums are only applied to sets of positive numbers), so the previous restriction does not apply here.

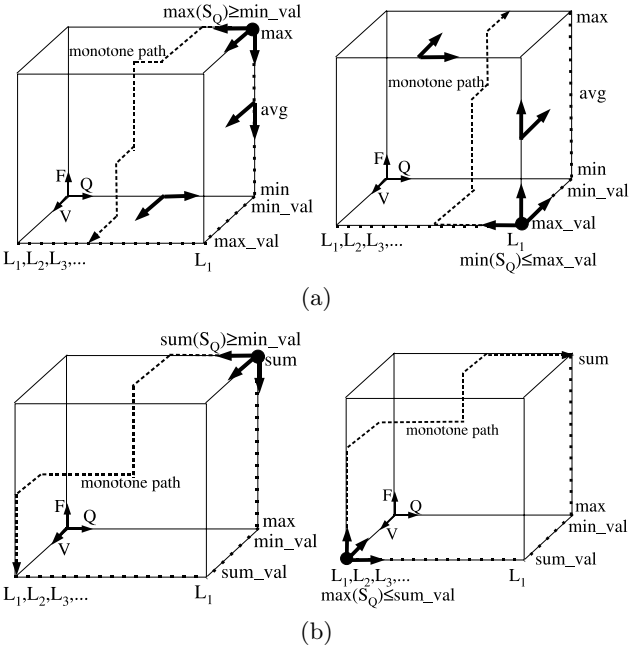


Fig. 1. (a) The refinement cubes for the generalized averages. (b) The refinement cubes for the generalized sums.

Remark that, when stretching the V-axis, the bottom face of the cubes can be connected to the top face of the cubes for the generalized averages, resulting in a combined refinement space.

The Refinement Cubes for the Generalized Counts. For the generalized counts the F-axis only contains the functions *count* and *count_distinct*. The V-axis ranges from 0 to the maximum size of the set generated by Q (*cnt_value*). The start conditions are $count(S_Q) \geq 0$ and $count\_dist(S_Q) \leq cnt\_value$. The monotone paths are the same as those for the generalized sums (see Fig. 1(b)).

Example. We now illustrate the use of the refinement cubes with an example. Consider the following start condition for the generalized sums:

$person(P, pos) \leftarrow sum(A, (child(P, C), age(C, A)), R), R \geq 10.$

Suppose we only use the functions *sum* and *max*, only use the threshold values 10 and 15, and only add a literal $male(C)$ to the query. Figure 2 schematically shows the refinements that are generated. Note that an aggregate condition can be obtained via more than one path, so in practice one has to take care to generate the conditions only once.

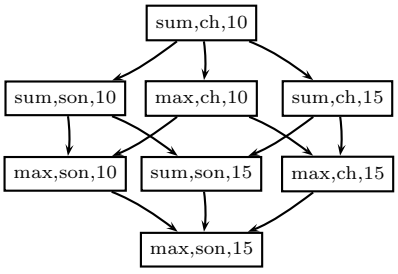


Fig. 2. Refinements generated for the start condition $max(\{A|child(P,C), age(C,A)\}) \geq 10$ by the cube for the generalized sums. In each node, the function F, query Q, and threshold value v are shown. The query is abbreviated: *ch* stands for $(child(P,C), age(C,A))$ and *son* means $(child(P,C), age(C,A), male(C))$.

Summary. Using the refinement cubes we obtain a search strategy that is

- *efficient*: only valid refinements are generated in each step, which allows to prune the aggregate search space,
- *complete*: every aggregate condition is reachable in a finite number of steps starting from the start conditions.

Remark that this efficiency can only be achieved by considering all three dimensions F, Q, and V together. A system that does not allow refinements along the F-axis (e.g., $max \rightarrow avg \rightarrow min$) can not obtain the aggregate conditions in a monotone (general-to-specific) way. For example, the condition $avg(\{A|child(P,C),\ age(C,A), male(C)\}) > 10$ can only be obtained via the monotone path $max(\{A|child(P,C), age(C,A)\}) > 10 \rightarrow max(\{A|child(P,C), age(C,A), male(C)\}) > 10 \rightarrow avg(\{A|child(P,C), age(C,A), male(C)\}) > 10$.

4 Application

In the previous section we have presented a general framework towards refining aggregate conditions. The framework can be beneficial for any relational learning system that learns aggregates and makes use of a general-to-specific ordering of the hypotheses to guide the search (e.g., decision tree learners, rule learners, frequent pattern miners,...). We now illustrate its usefulness by applying it to the relational decision tree learner Tilde [13] and showing that it yields a significant, though still modest, efficiency gain.

4.1 Tilde

Tilde is a relational decision tree learner. It learns trees with a divide and conquer algorithm similar to C4.5 [14]. The main point where it differs from the latter is that the tests to be considered at a node are Prolog queries. To split a

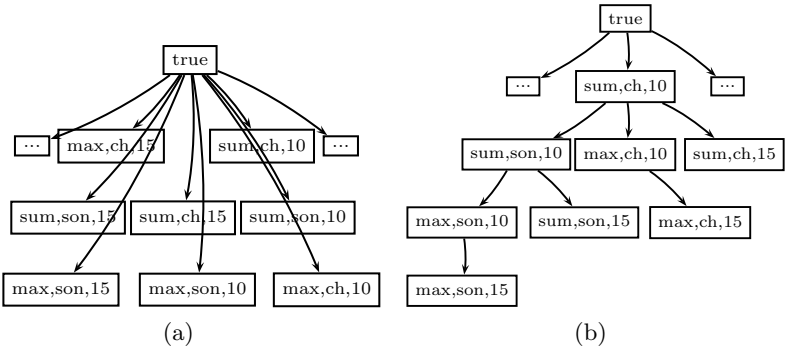


Fig. 3. (a) Original search space structure. (b) Optimized search space structure.

node, a refinement operator generates all allowed tests using a given hypothesis language specification. For each example corresponding to the node all tests are executed[2]. The test yielding the highest information gain is chosen.

Van Assche et al. [9] described how to add aggregate conditions to the hypothesis language of TILDE. Since the generality order on aggregate conditions was insufficiently understood, no structure was imposed on the aggregate search space, i.e., the tests in it were executed in random order. However, using the monotonicity properties discussed here, we know that whenever a test T fails for an example, none of the tests that can be obtained from T via monotone paths of the refinement cubes has to be executed against that example. Exploiting this knowledge would obviously result in an efficiency gain, while the same search space as before would be searched, and hence, the same tree would be obtained.

4.2 Experiments

We re-implemented TILDE's refinement operator in such a way that it structures the aggregate search space following the monotone paths of the refinement cubes[3]. Figure 3 shows how the search space for the example from Fig. 2 would be organized, both for the original and for the optimized refinement operator. If the test $sum(A, (child(P, C), age(C, A)), R), R \geq 10$ fails for an example, with the optimized method we could prune all its children in the search space, whereas in the original unstructured space, they would still all be tested.

In our experiments we compare the induction times obtained using the optimized refinement operator to those obtained when the original approach is used. In order to get the same set of tests at each node, we used the TILDE-LA setting [9] for the original refinement operator, where a lookahead of depth one is used

[2] This corresponds to the "examples in outerloop" approach (see [15]).

[3] In TILDE a query can only be refined by specializing it, not by dropping literals. Thus, in the context of the refinement cubes, refinements along the Q-axis can only take place in one direction.

Table 1. Runtime comparison of TILDE's optimized and original refinement operator

DATA SET	SEARCH	OPTIMIZED (SEC)		ORIGINAL (SEC)		SPEEDUP_RATIO	
	SP. SIZE	TOTAL	EXEC	TOTAL	EXEC	TOTAL	EXEC
MUTAGENESIS	53086	3792.54	2406.24	12079.97	10775.26	3.19	4.48
FINANCIAL	34900	4854.50	4717.08	12886.53	12751.98	2.65	2.70
DITERPENES	30533	12037.56	10518.78	16081.92	14535.39	1.34	1.38

in the aggregate query. For our comparison we used three datasets well-known in the ILP community: Mutagenesis [16], Financial [17], and Diterpenes [18].

For each of the experiments we used the aggregate functions *count*, *count distinct*, *max*, *avg*, and *min*. Table 1 gives an overview of the results we obtained[4]. The total runtime is reported, as well as the total time needed for executing the tests (summed over all nodes). Also the maximum size of the search space at a node in the tree is included. A first observation when looking at the table is that with the new method, a speedup factor of up to 4.5 is achieved for executing the tests. Second, since a large proportion of the total runtime goes into query execution, a significant overall runtime speedup is accomplished.

5 Conclusions and Future Work

The contributions of this paper are twofold. First, we have presented an in-depth study of the monotonicity of aggregate conditions, which is useful for refining models with aggregates. We have identified three dimensions along which monotonicity properties of an aggregate condition can be investigated: the aggregate function, the set to be aggregated, and the threshold value. This first dimension has never been explored before, but turns out to be crucial to obtain an efficient refinement strategy for aggregate conditions. Second, this derived strategy was applied to an existing relational decision tree learning system, illustrating the efficiency gain that can be achieved by using the results from this study.

An obvious direction for further work is to apply the framework to other relational systems. Another idea for future research is to search for aggregate function classes that contain other aggregate functions than the set we used.

Acknowledgements

Celine Vens is supported by the GOA 2003/8 project and by the Fund for Scientific Research (FWO) of Flanders. Jan Ramon and Hendrik Blockeel are postdoctoral fellows of the Fund for Scientific Research (FWO) of Flanders.

[4] Since accuracy, interpretability,... are unaffected by using this optimized refinement operator, they are not repeated here. We refer the reader to [9] for details.

References

1. Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained associations rules. In: SIGMOD International Conference on Management of Data. (1998) 13–24
2. Krogel, M.A., Wrobel, S.: Transformation-based learning using multi-relational aggregation. In: Proceedings of the 11th International Conference on Inductive Logic Programming. (2001) 142–155
3. Knobbe, A., de Haas, M., Siebes, A.: Propositionalisation and aggregates. In: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, Springer (2001) 277–288
4. Neville, J., Jensen, D., Friedland, L., Hay, M.: Learning relational probability trees. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2003)
5. Koller, D.: Probabilistic relational models. In: Proceedings of the 9th International Workshop on Inductive Logic Programming, Springer-Verlag (1999) 3–13
6. Perlich, C., Provost, F.: Aggregation-based feature invention and relational concept classes. In: Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press (2003) 167–176
7. Krogel, M.A., Wrobel, S.: Facets of aggregation approaches to propositionalization. In: Proceedings of the Work-in-Progress Track at the 13th International Conference on Inductive Logic Programming. (2003) 30–39
8. Knobbe, A., Siebes, A., Marseille, B.: Involving aggregate functions in multi-relational search. In: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery, Springer-Verlag (2002) 287–298
9. Van Assche, A., Vens, C., Blockeel, H., Džeroski, S.: First order random forests: Learning relational classifiers with complex aggregates. Machine Learning, Special Issue on ILP (2006), to appear
10. Uwents, W., Blockeel, H.: Classifying relational data with neural networks. In: Proceedings of 15th International Conference on Inductive Logic Programming, Springer (2005) 384–396
11. Muggleton, S., ed.: Inductive Logic Programming. Academic Press (1992)
12. Plotkin, G.: A note on inductive generalization. Machine Intell. **5** (1969) 153–163
13. Blockeel, H., De Raedt, L.: Top-down induction of first order logical decision trees. Artificial Intelligence **101**(1-2) (1998) 285–297
14. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann (1993)
15. Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G., Ramon, J., Vandecasteele, H.: Improving the efficiency of Inductive Logic Programming through the use of query packs. Journal of Artificial Intelligence Research **16** (2002) 135–166
16. Srinivasan, A., King, R., Bristol, D.: An assessment of ILP-assisted models for toxicology and the PTE-3 experiment. In: Proceedings of the 9th International Workshop on Inductive Logic Programming, Springer-Verlag (1999) 291–302
17. Berka, P.: Guide to the financial data set. In: The ECML/PKDD 2000 Discovery Challenge. (2000)
18. Džeroski, S., Schulze-Kremer, S., Heidtke, K.R., Siems, K., Wettschereck, D., Blockeel, H.: Diterpene structure elucidation from ^{13}C NMR spectra with inductive logic programming. Applied Artificial Intelligence **12**(5) (1998) 363–384

Measuring to Fit: Virtual Tailoring Through Cluster Analysis and Classification

Herna L. Viktor[1], Eric Paquet[2], and Hongyu Guo[1]

[1] School of IT and Engineering, University of Ottawa,
800 King Edward Road, K1N 6N5 Ottawa, Canada
{hlviktor, hguo028}@site.uottawa.ca,
http://www.site.uottawa.ca/~hlviktor/
[2] Visual Information Technology Group, National Research Council of Canada
50 Montreal Road, K1N 6N5 Ottawa, Canada
eric.paquet@nrc-cnrc.gc.ca

Abstract. Clothes should be designed to tailor well, fit the body elegantly and hide obvious body flaws. To attain this goal, it is crucial to know the interrelationships between different body measurements, such as the interplay between e.g. shoulder width, neck circumference and waist. This paper discusses a study to better understand the typical consumer, from a virtual tailor's perspective. Cluster analysis was used to group the population into five clothing sizes. Next, multi-relational classification was applied to analyze the interplay between each group's anthropometric body measurements. Throughout this study, three-dimensional (3-D) body scans were used to verify the validity of our findings. Our results indicate that different sets of body measurements are used to characterize each clothing size. This information, together with the demographic profiles of the typical consumer, provides us with new insight into our evolving population.

1 Introduction

Designing clothes that fit the population well is an important issue in the industry. This is of crucial importance, not only for the mass market, but also for expensive designer labels. Consider today's highly competitive clothing market. If a clothing brand that sells, for example jeans, do not produce garments that tailor well, the customers will take their business elsewhere. Clothes must fit the population. Otherwise, they will not sell. This implies that the different sizes (e.g. small, medium, large) must correspond to real body shapes, in the sense that one or more archetypes should represent any individual member belonging to the cluster. Clothing designers cannot utilize averages, since, if one average many human bodies, one ends up with a blob that has no relation whatsoever with any human being.

This gives rise to a number of important questions. What is the typical profile of consumers' bodies? How do the body measurements within different clothing sizes interrelate? For example, what is the interplay between the arm length, waist size and shoulder width of a typical medium-size male? What are the interrelationships with the demographic profiles of our consumers, in terms of

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 395–406, 2006.
© Springer-Verlag Berlin Heidelberg 2006

income, education, number of children, and age, amongst others. We may want to consider, e.g., the hip circumference, thigh circumference and bottom to knee height of the members of the female population with a high income, in order to design expensive pants to fit these consumers well. This paper discusses our experimental results when attempting to answer these questions. Here, we discuss how we analysed the body measurements of a number of human subjects, as contained in the CAESAR^{TM} anthropometric database. We verified the outcome of our results against a set of corresponding 3-D body scans, i.e. the scans of the subjects who are sitting and standing. This approach is used in order to ensure that our results correspond to reality.

This paper is organized as follows. Section 2 introduces the CAESAR^{TM} database, an anthropometric repository. Next, in Section 3, we introduce the experimental approach we followed to analyze this data. Section 4 contains the experimental results. This is followed, in Section 5, by our analysis of the results and the lessons learned. Section 6 concludes the paper.

2 CAESAR^{TM} Database

What is anthropometry? Anthropometry is the study of human body measurements (height, weight, size, proportions, etc.) and its biomechanical characteristics, including the stature, sizes of body parts and the space in which the body functions, e.g. reach its limits and clearances needed for movement. Anthropometric data therefore refer to a collection of physical dimensions of a human body. The aim of anthropometry is to characterize the human body by a set of measurements [1,2]. The field of anthropometry has many applications, ranging from the architectural and interior design of restricted spaces such as airline and vehicle seats, to the design of spaces for easy access by the elderly and the disabled. Anthropometry is also very important in retail tailoring, in order to ensure that the clothing fits the person well, is esthetically pleasing and comfortable. This application is the focus of our research, i.e. to investigate anthropometric body measure m! ents from a tailoring point of view. To this end, we are analyzing the CAESAR^{TM} database, as discussed next.

The CAESAR^{TM} Project is an international anthropometric survey that was carried out in the United States, Italy, the Netherlands and Canada. This survey involved a large number of individuals in each country. For each individual, a series of highly accurate anthropometric measurements was performed and questions of demographic nature were recorded. The anthropometric measurements included forty-nine details which have been recorded by domain experts, using standard anthropometric practices [1]. These include the stature, weight, thigh circumference, acromial height, feet length, and so on. The demographic data corresponding to the answers of the participants to questions regarding their family income, number of children, age, etc.

What makes the originality of this survey is that each person was scanned in three dimensions using a full body scanner. Different types of technologies were used but they are all built around the same principles. A few laser scanners move

along a rigid frame and capture a full body scan in a few seconds. Within the CAESAR^{TM} Project, each subject was scanned in three different postures. The 3-D body scans were described using a global shape-based descriptor, which is an abstract and compact representation of the three-dimensional shape of the corresponding body [1,2]. In essence, each scan was represented by a set of three histograms, which constitutes a 3-D shape index for the human body. Interested readers are referred to [2] for a detailed description of our cord-based approach.

This database thus represents a detailed and accurately measured subset of the typical human morphology, together with a 3-D body scan which maps these measurements and demographic profiling information.

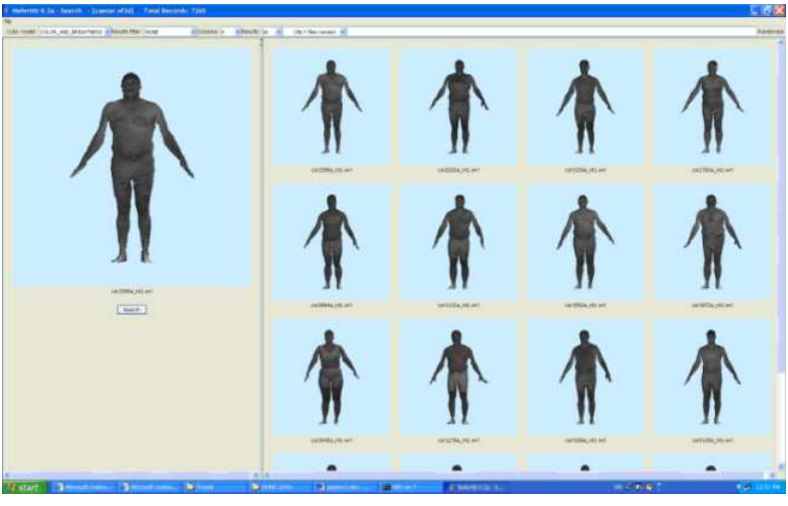


Fig. 1. An example of 3-D human body scans as contained in the CAESAR^{TM} database

3 Methodology

This section describes the experimental approach we followed to analyze the CAESAR^{TM} data. This data resides in an IBM DB2 relational database and our aim was to utilize this structure when mining the data *directly*, using multi-view learning.

Initially, the data was grouped into clusters based on the anthropometric body measurements. The aim here was to create clusters that correspond to typical industry-based clothing sizes, such as small, medium, etc. That is, to identify the natural body size groupings within the CAESAR^{TM}, in which intra-cluster similarity is high and inter-cluster similarity is low [3,4].

Next, this clustering information was used to determine the class labels, based on the body (or clothing) size. Figure 2 shows the Entity-Relationship (ER) diagram of the CAESAR^{TM} database. The figure shows that the database consists

of six (6) relational tables, with the BodySize class label as an attribute in the *Top_measurements* table. The database also contains tables describing the demographics of the human subjects, including the financial information, perceived fitness and body size, age, education and so on. This relational database was used for multi-view relational classification, as discussed next.

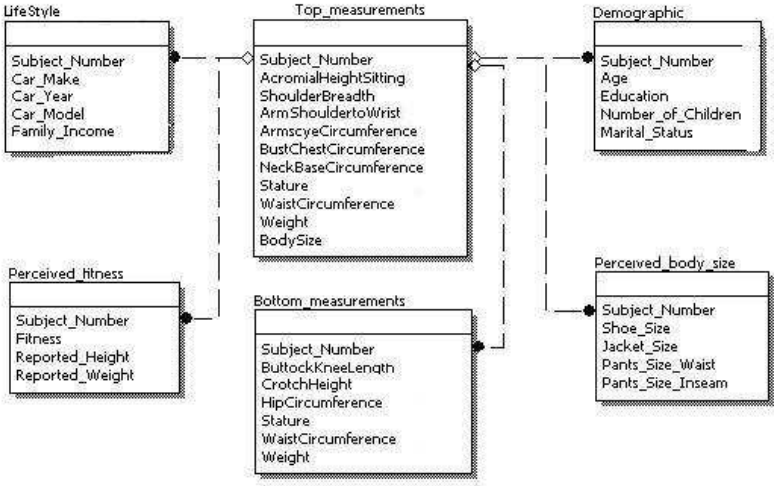


Fig. 2. The Entity Relationship (ER) diagram for the CAESAR^{TM} database

3.1 Multi-view Relational Classification

Recall that the CAESAR^{TM} data resides in a commercial IBM DB2 relational database. Our goal is to *directly* and *transparently* mine the data, without flattening or propositionalizing the database into a universal relation [5,6]. Our motivation is as follows. It usually takes considerable time and effort to convert relations into the required flat format. Another drawback is that this process can cause a loss of information [5] and can create a large amount of redundant data. Furthermore, a universal relation with a large amount of entities and attributes often leads to efficiency and scaling challenges. We aim to provide the clothing designers and anthropometric domain experts, who are not necessarily data miners, with easy, intuitive access to the CAESAR^{TM} database. Also, we envisage that the end results of our research presented here will, in future, be applied to much larger anthropometric databases with complex schemas. To this end, we engaged in multi-view learning, as discussed next.

What is multi-view learning? In the multi-view setting, each view of the data can be expressed in terms of a set of disjoint features of the training data. The target concept is learned independently, i.e. on each of these separate views using the features present. As in the example given by Blum and Mitchell [7], one can classify segments of televised broadcast based either on the video or on the audio information.

When considering a relational database, each relation (or table) thus corresponds to a view. One of these tables are considers as the target, while the others act as background relations. Each relation usually has a naturally divided disjoint feature set. When considering relational data mining, each of the relations therefore has diverse set of attributes, each providing different viewpoint (or "views") regarding the target concepts to be learned. This is especially evident when considering the CAESAR^{TM} database. Here, each relation describes different characteristics of a specific person. For example, the *Demographic* relation contains a person's demographic information, but the *Top-measurements* and *LifeStyle* relations identify a person's body measurements, income and automobile details. In other words, each relation from this database provides different types of information, or views, of the person.

Formally, in a relational classification setting, we have a database DB and a target relation T_{target} which includes a target variable y. The relational classification task is to find a function $F(x)$ which maps each tuple x of the target table T_{target} to the category y.

$$y = F(x, DB, T_{tar}) | x \in T_{tar}$$

In the above scenario, the T_{tar} table is considered the target relation while all others are background relations.

Our multi-view learning approach consists of two stages, firstly, an **Information Propagation** stage and, secondly, a **Multiple View Learning** stage [8]. During Information Propagation, the training data sets of the multiple views are constructed. Here, each background relation obtains the classes from the target relation. Aggregation information, if any, is also propagated. (Note that the CAESAR^{TM} database denotes a special case, with one-to-one relationships between tables.) Next, each of the views is used to construct various hypotheses on the target concept, using traditional single-table learning algorithms.

Let us again consider the construction of the view for the *Demographics* table. During the **Information Propagation Stage**, both relations *Top-measurements* and *Demographics* have the class labels to be learned. Thus, each of them is *individually* able to construct a target function for the target concept. However, the relation *Top-measurements* only includes information about a person's body measurements. Therefore hypothesis produced by this relation will only reflect how those body measurements such as Stature and Weight relate to the class label, namely, BodySize. On the other hand, the *Demographics* relation only consists of information about a person's demographic information. Consequently, classification model built using this relation will only have the relationship between class BodySize and people's demographic information, such as Age, Education and Marital Status, etc.

Note that a detailed discussion of our multi-view learning method falls beyond the scope of this paper. Interested readers are referred to [8] of an overview of our approach.

4 Experimental Results

We implemented the experiments using the WEKA data mining system, a Java-based knowledge learning and analysis environment developed at the University of Waikato in New Zealand [9]. The Cleopatra multimedia information retrieval system was used to verify the quality of the end results, against the 3-D body scans [1]. Recall that the CAESAR^{TM} database was implemented in IBM DB2.

The original data set consisted of the data of 670 human subjects, each containing one 3-D body scan of a person facing the scanner and 15 demographic attributes, together with 49 anthropometric attributes for females and 48 anthropometric attributes for males. (Note that no under bust circumference of the male subjects was recorded.) The data was separated into two sets based on the gender of the subject, to form two sets of 414 males and 256 females, respectively. The results when exploring the data regarding the 414 male subjects are reported here. The subjects we analyzed were all residents of the USA, thus making them a pool of similar genetic and environmental conditions.

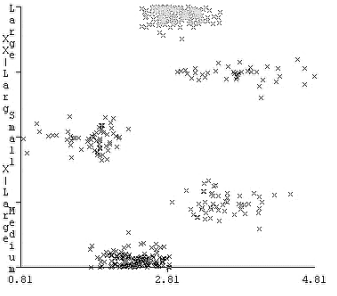


Fig. 3. Visualization of Clusters

4.1 Cluster Analysis

Firstly, the anthropometric data was considered in order to identify typical clusters within the male population. A number of clustering algorithms was considered. These included partitioning, hierarchical, density-based, model-based and grid-based approaches [3,4]. We also considered the use of principal component analysis (PCA) to reduce the number of attributes prior to clustering.

In our experiments, the number of initial clusters was set to five (5), since that corresponds to the number of clothing sizes we were interested in identifying. By inspection, through the analysis of the results using the Cleopatra and Weka systems, the k-means classifier algorithm was found to produce the best clusters. Also, it was found that there where no need to apply PCA prior to clustering, since the inter-cluster relationships were weak. The k-means algorithm is a Centroid-based partitioning technique, in which each cluster is represented by the mean value of the objects in the cluster and the Centroid is viewed as

the cluster's centre of gravity [3,4]. This technique is highly suitable for discovering similar sized clusters with convex shapes, on numeric attributes. Also, it works well with domain with little noise or outliers. The body measurements were recorded and verified by at least two anthropometric domain experts, thus making the level of ! noise and occurrence of outliers rare.

Table 1. Body measurements of cluster Centroids as contained in the CAESAR^{TM} database.

	Small	**Medium**	**Large**	**X-Large**	**XX-Large**
BustChestCircumference	94.9(5.9)	99.4(6.8)	106.1(6.19)	106.9(6.6)	125.7(10.9)
WaistCircumference	82.0(6.8)	86.8(6.8)	92.3(7.5)	107.8(7.0)	116.6(16.0)
HipCircumference	96.70(4.80)	101.14(4.93)	106.48(5.42)	108.39(4.90)	123.15(14.33)
NeckBaseCircumference	44.6(1.9)	46.0(2.0)	48.0(2.19)	48.7(2.0)	52.7(3.04)
ArmShouldertoWrist	59.5(2.03)	62.4(1.79)	65.08(1.71)	68.7(1.96)	65.56(3.02)
Stature	167.0(5.9)	174.7(3.5)	180.6(3.6)	190.0(5.0)	181.9(4.6)
ShoulderBreadth	44.65(1.90)	45.96(2.03)	48.00(2.19)	48.69(2.00)	52.66(3.04)
Weight(lbs)	151.17(15.77)	172.19(17.70)	198.16(18.61)	212.85(20.04)	274.99(35.9)
Num. of Members	62 (15%)	131(31%)	131 (31%)	52 (13%)	38 (9%)

For the male population, the Centroids of the clusters are depicted in Table 1. The distribution of the clusters is depicted in Figure 3. As can be seen from Figure 3, the cluster sizes are comparable and the shapes thereof are convex, therefore making k-means a good choice.

We subsequently verified the cluster membership through querying the 3-D body scans using the Cleopatra system. Table 1 shows some of the characteristics of the Centroids of the male population [10]. Shown are the means (in cm) and standard deviations of each of the clusters and the number of cluster members. Figure 4 shows the 3-D body scans of the human subjects that correspond to these anthropometric measurements, highlighting the difference in body types of the five clusters. The clusters as also depicted in Figure 3. From Table 1 and Figures 3 and 4, and our manual inspection, it follows that the anthropometric clusters distinguish between small, medium, large, extra-large and extra-extra-large body sizes.

4.2 Multi-view Relational Classification

Next, we engaged in multi-relational classification, using the five clusters of body size, as class labels. For our multi-relational experiments we used the benchmarking C4.5 decision tree algorithm [11], the PART rule learner that constructs decision list based on the repeated creation of partial decision trees [12] as well as the widely used RIPPER propositional rule inductor, which have shown to generate rules that are easy for humans to understand [13,14].

We proceeded to analyze the data in order to find the rules to describe the body measurements within each class. To this end, we executed our multi-view learning

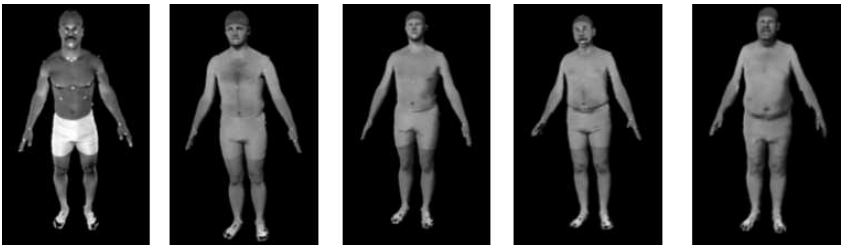


Fig. 4. Cluster Centroids for Male Population

Table 2. Accuracy and number of rules for three multi-view learning tasks

Learning task	Ripper	C4.5	PART
Top measurements	78.7% (10)	78.2% (30)	76.6% (18)
Bottom measurements	76.3% (11)	78.2% (19)	77.5% (19)
Top and Bottom measurements	79.2% (12)	80.1% (30)	78.7% (21)

algorithm three times, used the (1) *Top_measurements*, (2) *Bottom_measurements* and (3) *Top_* and *Bottom_measurements* relations as target relations.

Table 2 shows the accuracies of the three learners against the three different learning tasks, for C4.5, RIPPER and PART. The table shows that the learning approaches provided consistent results, in terms of accuracy and number of rules constructed. Next, we identified those rules with high positive and little negative coverage. This makes sense from an application point of view, since we aimed to obtain interesting, general rules for application in industry. Table 3 shows some of the most important rules we found for each body size, in terms of the information given and the rule coverage. Shown are the positive and negative coverage, as well as the percentage of subjects within that class which satisfies each rule.

When considering the rule set in Table 3, the following observations are noteworthy. The table shows that, for the extra-extra large size, the rules mainly address the weight and stature. Again, this makes sense from an application point of view. Extra-extra large individuals are often difficult to characterize, due to the nature of their body shape and the difficulty to obtain accurate measurements. This fact makes it difficult for tailors to design clothing that fits these individuals well, as mentioned by many of the participants in this study. From the rule set, one can, in addition, conclude that when designing clothes, the attributes to be taken into account are different for each clothing size. For example, for the large clothing size the width of the shoulders and the length of the arms are important factors. Our rules also indicate that Small (or Thin) individuals are usually shorter than the other body sizes. This implies that, for this group, the majority ! of e.g. pants should have short lengths. Otherwise, a mass market manufacturer may be left with an overflow of pants that cannot sell.

Table 3. Some high coverage rules for each clothing/body size

	Class	Coverage	Positive Coverage(%)	Learner
Learning Top Measurements				
(Stature ≤ 170.9) AND (BustChestCircumference ≤ 100.2)	Small	(44/ 3)	71.0	RIPPER
(167 < Stature ≤ 176) AND (163 < Weight ≤ 213.5) AND (ArmShouldertoWrist ≤ 65.0)	Medium	(77/5)	58.8	C4.5
(Weight > 179) AND (Stature ≤ 182.5) AND (ArmShouldertoWrist > 60.4) AND (ShoulderBreadth ≤ 54.4)	Large	(73/1)	55.7	PART
(Stature > 184.2) AND (Weight < 243) AND (ArmShouldertoWrist > 67.9)	X Large	(31/0)	59.6	C4.5
(Weight ≥ 241)	XX Large	(38/5)	100.0	RIPPER
Learning Bottom Measurements				
(Stature ≤ 170.9) AND (Weight ≤ 190)	Small	(52/3)	83.8	RIPPER
(167 < Stature ≤ 176) AND (Weight > 163) AND (HipCircumference ≤ 113.5) AND (ButtockKneeLength ≤ 63.5)	Medium	(78/6)	59.5	C4.5
(176<Statute≤182.5) AND (183<Weight≤243) AND (CrotchHeight ≤ 87)	Large	(72/5)	55.0	C4.5
(Stature≥184.3) and (WaistCircumference≥93.2)	X Large	(35/5)	67.3	PART
(Stature > 176) AND (Weight > 243) AND (CrotchHeight ≤ 84.3)	XX Large	(29/0)	76.3	C4.5
Learning Top and Bottom Measurements				
(Stature ≤ 171.9) AND (HipCircumference ≤ 98.1)	Small	(36/1)	58.0	RIPPER
(167<Stature≤176) AND (163<Weight≤199.5) AND (HipCircumference ≤ 113.5) AND (ButtockKneeLength ≤ 63.5) AND (CrotchHeight > 73)	Medium	(61/0)	46.5	PART
(183 < Weight ≤ 243) AND (62 < ArmShouldertoWrist ≤ 66.9) AND (CrotchHeight≤87) AND (Stature>176)	Large	(94/8)	71.8	C4.5
(ButtockKneeLength > 61.4) AND (BustChestCircumference ≤ 113)	X Large	(31/5)	59.7	PART
(Weight ≥ 239) and (BustChestCircumference ≥ 118.4)	XX Large	(28/0)	73.7	RIPPER

The positive coverages of the rules shown here indicate that they are general enough to be applied when tailoring clothes. Their number is sufficient to constrain the design without increasing the production costs. Indeed, the later is increased if too many rules or constraints need to be implemented.

Note that we also analyzed the data within each class to better understand the demographic nature of our customer. When considering the typical profiles of the individuals who correspond to the individual clothing sizes, we were able to pinpoint interrelationships between fitness, family income and education, amongst others. For example, our exploration of the demographic profiles of the

medium-sized males indicate that, for this data set, a 25 to 40 aged single male with a high level of fitness who holds a Bachelors degree earns, on average, substantially more than an unfit person with the same background. Also, a married person with a doctorate earns more than his single counterpart, irrespective of his marital status. Due to limited space, interested readers are referred to [15] for a detailed discussion.

5 Analysis of Results

Clothes must fit the population. If they do not, they will not sell. This implies that the different sizes (e.g. small, medium, large) must correspond to real groupings within the population. Each cluster must be well defined in the sense that one archetype can represent any individual member belonging to the cluster. Clothing designers cannot utilize averages. This is not difficult to understand, if one average many human bodies, one ends up with a blob that has no relation whatsoever with any human being.

Consequently, it is important to define clusters that can be characterized by one archetype, i.e. a representative individual that belongs to the cluster and that represent any other member of the cluster. Usually this archetype corresponds to the closest individual to the Centroid of the cluster. Such a working hypothesis is based on the implicit assumption that the cluster has a spherical or quasi spherical symmetry. One can also choose one of the individuals belonging to the sub-region presenting the highest density in terms of number of individuals. If it is not the case, more than one archetype might be necessary to fully characterize the cluster. In practice, it is preferred to have only one. Each new size of sub-size involves more costs and increase the complexity involved in both the manufacturing and the distribution process.

It is also important that each cluster represent a large proportion of the population. If it is not the case, clothes are manufactured for a size that fits virtually nobody which again implies financial lost. Nevertheless, clusters with a small number of members can be justified for very expensive clothes for which a good fit is a paramount condition.

The method we utilised satisfies the above mentioned requirement because we were able to group the individuals into one of five clusters with a well-defined Centroid. Our verification, by means of the Cleopatra system, indicated that the cluster membership do correspond to the reality, in the sense that the bodies did correspond to our expectations of the cluster membership.

Also, we were able to create sets of rules that describe the interrelationships between the different body measurements within each of the five groupings. Importantly, the rules indicate which of these measurements need to be taken into account when tailoring clothes. For example, for the medium-sized individuals, the rules indicated that the Stature, Hip Circumference, Buttock to Knee Length and Crotch Height are important when designing pants.

When exploring the clusters as class labels, we used three different classification methods, which gave us consistent sets of rules, in terms of overall accuracy,

as well as coverage. We have discovered some worth-while relationships between this data. For example, it is interesting to note that, when e.g. distinguishing between large and extra large subjects within the male population, the chest and hip circumferences do not differ substantially. Rather, the waist circumference and stature is of importance here. This reality is not reflected in the "ideal" clothing sizes and is thus an important factor to consider when designing clothes. One of the most remarkable results that we have obtained is that the attributes that must be considered when designing clothes differ according to clothing size.

6 Conclusion

Tailoring clothes that fit the population poses an important challenge to the clothing industry. The mining of anthropometric body measurements, in order to understand and future explore the body profile of the typical consumer has application in the mass marketplace as well as in haute couture both for the design and the distribution. Such an understanding is fundamental in order to minimize the number of returned items in e.g. Internet shopping.

This paper discussed our results when analyzing a data set containing anthropometric body measurements, together with demographic data. We were able to cluster the population into five well-defined clusters. We also generated some general rules with high coverage, for application in industry. Our results showed that the body measurements to be taken into account for the various body sizes differ substantially. That is, each clothing size is characterised by a different set of anthropometric body measurements. We were able to verify the results of our experimentation against a set of corresponding 3-D body scans. Future work will include the analysis of the 3-D body scans to complement the results presented in this paper. Our preliminary analysis, when clustering the 3-D body scan histograms, indicate that clusters, which complement those presented here, can be found [16].

References

1. Paquet, E., Robinette, K.M., Rioux, M.: Management of three-dimensional and anthropometric databases: Alexandria and cleopatra. Journal of Electronic Imaging **9**(4) (2000) 421–431
2. Brunsman, M.A., Daanen, H.M., Robinette, K.M.: Optimal postures and positioning for human body scanning. In: NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Washington, DC, USA, IEEE Computer Society (1997) 266
3. Abdali, O., Viktor, H.L., Paquet, E., Rioux, M.: Exploring anthropometric data through cluster analysis. SAE 2004 Transactions Journal of Aerospace **113-1** (2004) 241–244
4. Dunham, M.H.: Data Mining: Introductory and Advanced Topics. Prentice Hall PTR, Upper Saddle River, NJ, USA (2002)
5. Dzeroski, S., Lavrac, N.: editors, Relational Data Mining. Springer, Berlin (2001)

6. Krogel, M.A., Wrobel, S.: Facets of aggregation approaches to propositionalization. In: Proceedings of the Work-in-Progress Track at the ILP. (2003)
7. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers. (1998) 92–100
8. Guo, H., Viktor, H.L.: Mining relational databases with multi-view learning. In: MRDM '05: Proceedings of the 4th international workshop on Multi-relational mining, New York, NY, USA, ACM Press (2005) 15–24
9. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques with Java implementations. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2000)
10. Viktor, H., Paquet, E.: A multi-strategy approach for mining multimedia data repositories, Data mining 2005. Skiathos: Greece, WIT Press: UK (2005)
11. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
12. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, San Francisco, CA, USA (1998) 144–151
13. Furnkranz, J., Widmer, G.: Incremental reduced error pruning. In: Proceedings of the 11th International Conference on Machine Learning (ML-94). (1994) 70–77
14. Cohen, W.W.: Fast effective rule induction. In Prieditis, A., Russell, S., eds.: Proc. of the 12th International Conference on Machine Learning, Tahoe City, CA, Morgan Kaufmann (1995) 115–123
15. Viktor, H.L., Paquet, E., Guo, H.: Demographic profiling of CAESAR^{TM} anthropometric data. Technical Report 001-2006, Intelligent Data Lab, School of IT and Engineering, University of Ottawa (2006)
16. Paquet, E., Viktor, H.L.: Anthropometric calibration of virtual mannequins through cluster analysis and content-based retrieval of 3-D body scans. In: Proc. of the 2005 IEEE Instrumentation and Measurement Technology Conference, Ottawa, Canada, IEEE Press (2005)

RIVA: Indexing and Visualization of High-Dimensional Data Via Dimension Reorderings

Michail Vlachos[1], Spiros Papadimitriou[1],
Zografoula Vagena[2], and Philip S. Yu[1]

[1] IBM T.J. Watson Research Center, Hawthorne, NY, USA
[2] IBM Almaden Research Center, San Jose, CA, USA

Abstract. We propose a new representation for high-dimensional data that can prove very effective for visualization, nearest neighbor (NN) and range searches. It has been unequivocally demonstrated that existing index structures cannot facilitate efficient search in high-dimensional spaces. We show that a transformation from points to sequences can potentially diminish the negative effects of the dimensionality curse, permitting an efficient NN-search. The transformed sequences are optimally reordered, segmented and stored in a low-dimensional index. The experimental results validate that the proposed representation can be a useful tool for the fast analysis and visualization of high-dimensional databases.

1 Introduction

Suppose that we are interested in performing search operations on a set of high-dimensional data. For simplicity let us assume that the data lie in a unit hypercube $C = [0, 1]^d$, where d is the data dimensionality. Given a query point, the probability P_w that a match (neighbor) exists within radius w in the data space of dimensionality d is given by $P_w(d) = w^d$. Figure 1(a) illustrates this probability for various values of w. Evidently, at higher dimensionalities the data becomes very sparse and even at large radii, only a small portion of the entire space is covered. This is also known as dimensionality curse, which in simple terms translates into the following fact: for large dimensionalities existing indexing structures outperform sequential scan only when the dataset size (number of objects) grows exponentially with respect to dimensionality.

In this work we propose a mapping from high-dimensional to low-dimensional spaces that will boost the performance of traditional indexing structures—such as R-trees—without changing their inner-workings, structure or search strategy. This mapping will essentially condense the sparse/unused data space by grouping and indexing together dimensions that share similar characteristics. We will accomplish this by applying the following transformations: i) Conceptually, we will treat high-dimensional data as ordered sequences. ii) The original D dimensions will be reordered to obtain a *globally* smooth sequence representation. This will lead to placement of dimensions with similar behavior at adjacent positions in the ordered representation as sequence. iii) The resulting sequences will

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 407–420, 2006.
© Springer-Verlag Berlin Heidelberg 2006

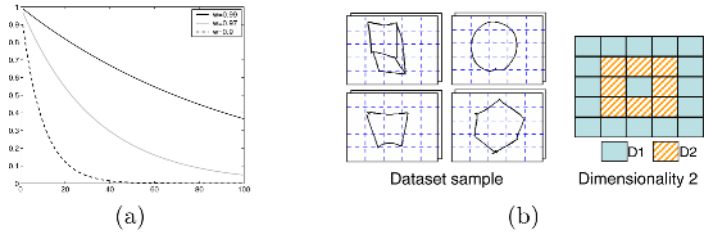


Fig. 1. (a) Probability $P_w(d)$. (b) Mapping of 25-D image features onto 2 dimensions and correspondence of projected against original dimensions.

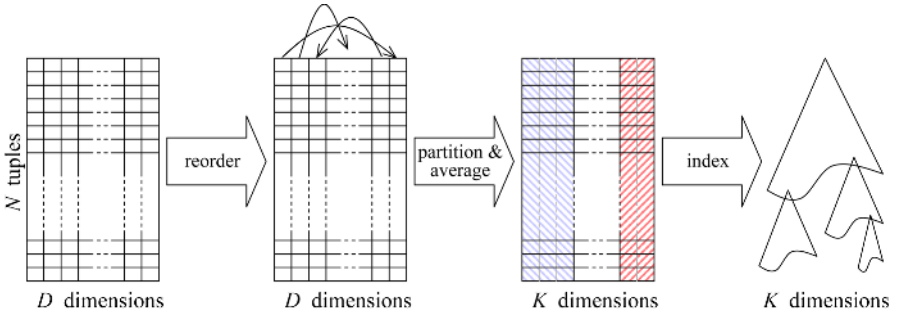


Fig. 2. Summarization of steps

be segmented into groups of $K < D$ dimensions which can be then stored in a K-dimensional indexing structure. The previous steps are illustrated in Figure 2.

Elements of our techniques are partially inspired by or adapted from concepts in parallel coordinates visualization [4], time-series representation [9], co-clustering and bi-clustering [8] methodologies. However, the final goal is distinct from the previous techniques, since the focus of this work is primarily on the indexing of high-dimensional data. We note, however, that since our approach also relies on the efficient grouping of correlated/co-regulated attributes, the proposed algorithms can also be utilized for the identification of the principal data axes for high-dimensional datasets.

In Figure 1(b) we demonstrate an example of the dimension grouping and dimensionality reduction achieved by our techniques. The dataset consists of 25-dimensional features extracted from multiple images using a 5×5 grid (more details are provided in the experimental section). Each image belongs to one of the following four shape classes: *cube, ellipse, hexagon and trapezoid*. The shapes are drawn by humans, so they exhibit dislocations or distortions and no two images are identical. Using the proposed low dimensional projection/grouping, we map each 25-dimensional point onto 2 dimensions. We depict the correspondence between sets of original dimensions and each of the projected dimensions. Observe that peripheral and center parts of the image (which correspond to almost empty pixel values) are collapsed together into one projected dimension. Simi-

Table 1. Description of main notation

Symb.	Description	Symb.	Description
N	Database size (number of points).	$\mathcal{D}$	An ordering of all D dimensions.
D	Database dimensionality.	K	Number of dimension partitions.
$\mathbf{t}_i$	Tuples (row vectors), $\mathbf{t}_i \in \mathbb{R}^D$.	$\mathcal{B}$	Set of partition breakpoints.
$t_i(d)$	The d-th coordinate of $\mathbf{t}_i$.	$\mathcal{D}_k$	The k-th ordered partition.
$\mathbf{T}$	Database, as a $N \times D$ matrix.	D_k	Size of $\mathcal{D}_k$.

larly centrally located portions of the image are also grouped together to form the second dimension. While this example illustrates the usefulness of our dimension grouping techniques for image/multimedia data, we should emphasize the utility of our methods in a number of other domains:

1. High-dimensional data visualization: Our technique intelligently groups of related dimensions, leading to an efficient low-dimensional interpretation and visualization of the original data. Our methods provide a direct mapping from the low-dimensional space to the original dimensions, permitting more coherent interpretation and decision making based on the low-dimensional mapping (contrast this with PCA, where the projected dimensions are not readily interpretable, since they involve translation and rotation on the original attributes).
2. Gene expression data analysis: Microarray analysis provides an expedient way of measuring the expression levels for a set of genes under different regulatory conditions. They are therefore very important for identifying interesting connections between genes or attributes for a given experiment. Gene expression data are typically organized as matrices, where the rows correspond to genes and columns to attributes/conditions. Our techniques could be used to mine either conditions that collectively affect the state of a gene or, conversely, sets of genes that are expressed in a similar way (and therefore may be jointly affecting certain variables of the examined disease or condition).
3. Recommendation systems: An increasing number of companies or online stores use collaborative filtering to provide refined recommendations, based on the historical user preferences. Utilizing common/similar choices between groups of users, companies like Amazon or Netflix can provide suggestions on products (or movies, respectively) that are tailored to the interests of each individual customer. For example, Netflix serves approximately 3 millions subscribers providing online rentals for 60,000 movies. By expressing rental patterns of customers as an array of customers versus movie rentals, our technique could be then used for identifying groups of related movies based on the historical feedback.

Summarizing the main contributions of this work: (i) We provide an efficient abstraction that can map high dimensional datasets into a low-dimensional space. (ii) The new space can be used to visualize the data on two (or three) dimensions. (iii) We demonstrate how the low dimensional space can be used in conjunction with existing indexing structures (such as R-trees) for mitigating the adverse effect of high-dimensionality on the index search performance.

(iv) Finally, the proposed data mapping effectively organizes the data features into logical subsets. This readily allows for efficient determination of correlated or co-regulated data features.

2 Preliminaries

In the following sections we will describe our methodology for data reorganization which is called 'RIVA' (**R**eordering for **I**ndexing and **V**isu**A**lization). Assuming a database $\mathbf{T}$ that consists of N points (rows) in D dimensions (columns), the goal is to reorder and partition the dimensions into K segments, $K < D$. We denote the database tuples as row vectors $\mathbf{t}_i \in \mathbb{R}^D$, for $1 \leq i \leq N$. The d-th value of the i-th tuple is $t_i(d)$, for $1 \leq d \leq D$. We begin by first defining an ordered partitioning of the dimensions. Then we introduce the necessary measures that characterize the quality of a partitioning, irrespective of order. Next, in Section 3 we will show how we can exploit reordering to find the partitions efficiently, with a single pass over the database.

Definition 1 (Ordered partitioning $(\mathcal{D}, \mathcal{B})$). *Let $\mathcal{D} \equiv (d_1, \ldots, d_D)$ be a* total ordering *of all D dimensions. The order along with a set of breakpoints $\mathcal{B} = (b_0, b_1, \ldots, b_{K-1}, b_K)$ defines an* ordered partitioning*, which divides the dimensions into K segments (by definition , $b_0 = 1$ and $b_K = D + 1$ always). The size of each segment is $D_k = b_k - b_{k-1}$. We denote by $\mathcal{D}_k \equiv (d_{k,1}, \ldots, d_{k,D_k})$ the portion of $\mathcal{D}$ from positions b_{k-1} up to b_k, i.e., $d_{k,j} \equiv d_{j-1+b_{k-1}}$, for $1 \leq j \leq D_k$.*

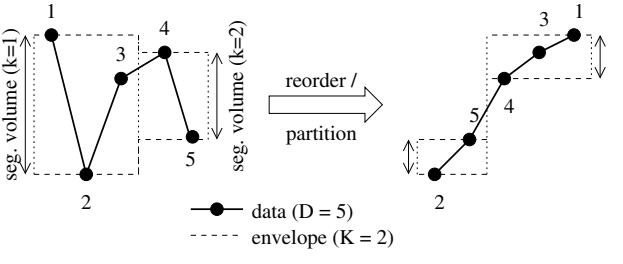


Fig. 3. K-dimensional envelopes of D-dimensional points. On the right, order is $\mathcal{D} = (2, 5, 4, 3, 1)$ with breakpoints $\mathcal{B} = (1, 3, 6)$ and partition sizes $D_1 = 2$ and $D_2 = 3$.

Next, we need a measure of quality. Given a partitioning, consider a single point $\mathbf{t}_i$. Ideally, we want the smallest possible variation among values of $\mathbf{t}_i$ within each partition $\mathcal{D}_k$. Figure 3 illustrates two different partitionings and their corresponding *envelopes* (dashed lines), which are simply the minimum and maximum values of $\mathbf{t}_i$ within each set of dimensions $\mathcal{D}_k$. The partition on the right has smaller *volume*.

Definition 2 (Envelope volume $v_i(\mathcal{D}, \mathcal{B})$). *The* envelope volume *of a point* $\mathbf{t}_i$, $1 \le i \le N$ *is defined by*

$$v_i(\mathcal{D}, \mathcal{B}) := \sum_{k=1}^{K} \big(\max_{d \in \mathcal{D}_k} t_i(d) - \min_{d \in \mathcal{D}_k} t_i(d)\big).$$

This is proportional to the average (over partitions) envelope width.

Definition 3 (Total volume $V(\mathcal{D}, \mathcal{B})$). *The* total volume *achieved by a partitioning is* $V(\mathcal{D}, \mathcal{B}) := \sum_{i=1}^{N} v_i(\mathcal{D}, \mathcal{B})$

We should point out that, although the width of an envelope segment $\mathcal{D}_k$ is related to the variance *within* that partition, the envelope volume v_i is different from the variance (over dimensions) of $\mathbf{t}_i$. Furthermore, the total volume V is not related to the vector-valued variance of all points, and hence is also not related to the per-column variance of $\mathbf{T}$, which is used in [2].

Summarizing, we shall seek a single partitioning of the dimensions for the entire database. To that end, we would like to minimize the total volume V.

3 Reordering

In the previous section we defined the notions of an ordered partitioning and of volume. Unfortunately, summation over all database points in V is the outermost operation. Hence, computing or updating the value of V would require buffer space kN for the minimum values and another kN for the maximum values, as well as $O(N)$ time. Since N is very large, direct use of V to find the partitioning is infeasible. Surprisingly, we can intelligently use the dimension ordering and recast the problem in a way that allows us to perform the search after a *single pass* over the database. The reordering of dimensions is chosen to maximize some notion of "aggregate smoothness" and serves two purposes: (i) provide an accurate estimate of the volume V without requiring $O(N)$ space and time, and (ii) locate the partition breakpoints. The next sections make these ideas precise.

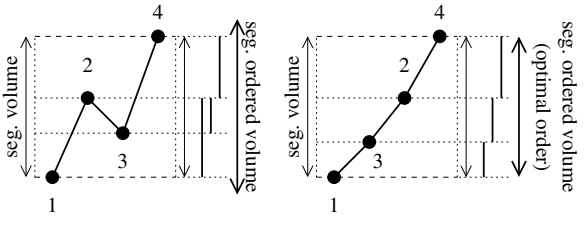


Fig. 4. Ordered volume for one data point, within a segment (see first segment in Figure 5), showing exactly the two volumes. Points on the right are in optimal order (see Lemma 1) and the ordered volume equals the "true" volume.

3.1 Volume Through Ordering

Consider a point $\mathbf{t}_i$ and a partition $\mathcal{D}_k$. Instead of the difference between the minimum and maximum over *all* values $t_i(d)$ for $d \in \mathcal{D}_k$, we will consider the sum of differences between *consecutive* values in $\mathcal{D}_k$.

Definition 4 (Ordered envelope volume $\bar{v}_i(\mathcal{D}, \mathcal{B})$). *The* ordered envelope volume *of a point* $\mathbf{t}_i$, $1 \leq i \leq N$ *is defined by*

$$\bar{v}_i(\mathcal{D}, \mathcal{B}) := \sum_{k=1}^{K} \sum_{j=2}^{D_k} |t_i(d_{k,j}) - t_i(d_{k,j-1})| = \sum_{\substack{j=1 \\ j \notin \mathcal{B}}}^{D} |t_i(d_j) - t_i(d_{j-1})|.$$

Figure 4 shows the ordered volumes of two different dimension orderings in one segment. The thin double arrows show the segment's volume. The thick lines on the right margin show the consecutive value differences. Their sum is the segment's ordered volume (thick double arrow).

Lemma 1 (Ordered volume). *For any ordering* $\mathcal{D}$*, we have* $v_i(\mathcal{D}, \mathcal{B}) \leq \bar{v}_i(\mathcal{D}, \mathcal{B})$*. Furthermore, holding* $\mathcal{B}$ *fixed, there exists an ordering* $\mathcal{D}^*$ *for which the above holds as an equality,* $\bar{v}_i(\mathcal{D}^*, \mathcal{B}) = v_i(\mathcal{D}, \mathcal{B})$*.*

The order $\mathcal{D}^*$ for which the ordered volume matches the original envelope volume of any point $\mathbf{t}_i$ is obtained by sorting the values of $\mathbf{t}_i$ in ascending (or descending) order—the full proof is ommited for space.

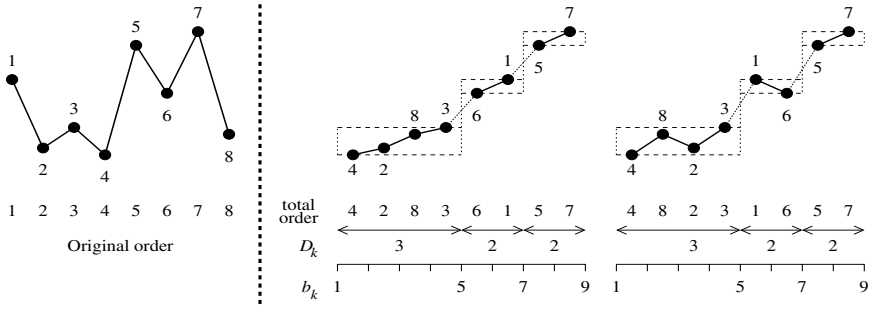


Fig. 5. One point and two total orders that correspond to the same partitioning ($D = 7$ and $K = 3$). The breakpoints b_k, $0 \leq k \leq K$ are also shown, with the induced partition sizes D_k, $1 \leq k \leq K$. The total ordering serves two purposes. First, to make the ordered volume within individual partitions close to the "true" volume. Second, to help us find the best breakpoints, which minimize the envelope and total volumes. The first reordering minimizes the sum of consecutive value differences, and achieves both goals.

Definition 5 (Total ordered volume). *The* total ordered volume *achieved by a partitioning is* $\bar{V}(\mathcal{D}, \mathcal{B}) := \sum_{i=1}^{N} v_i(\mathcal{D}, \mathcal{B})$*.*

Lemma 1 says that, for a given point $\mathbf{t}_i$, the ordering $\mathcal{D}$ allows estimation of the envelope volume using the sum of consecutive value differences. Furthermore, using a similar argument, we can show that a reordering $\mathcal{D}$ also helps us find

the best breakpoints for a single point, i.e., the ones that minimize its envelope volume (see Figure 5—proof is straightforward but long and ommite d for space).

Lemma 2 (Envelope breakpoints). *Let $\mathcal{D}^* \equiv (d_1, \ldots, d_D)$ be the ordering of the values of $\mathbf{t}_i$ in ascending (or descending) order. Given $\mathcal{D}^*$, let the breakpoints $b_1, \ldots, b_{K-1}$ be the set of indices j of the top-$(K$–$1)$ consecutive value differences $|t_i(d_j) - t_i(d_{j-1})|$ for $2 \leq j \leq D$. Then, $v_i(\mathcal{D}^*, \mathcal{B}^*) = \bar{v}_i(\mathcal{D}^*, \mathcal{B}^*)$ and this is the minimum possible envelope volume over all partitionings $(\mathcal{D}, \mathcal{B})$.*

3.2 Rewriting the Volume

Next, we show that optimizing for $\bar{V}$, instead of V, can be performed with only a single pass over the database. By substituting the minimum and maximum operations (in v_i) with a summation (in $\bar{v}_i$), it is possible to exchange the summation order and make the summation over all points the innermost one. This allows us to compute this quantity once, hence requiring only a single scan of the database. First, we give a name to this sum.

Definition 6 (Dimension distance). *For any pair of dimensions, $1 \leq d, d' \leq D$, their* dimension distance *is the L^1-distance between the d-th and d'-th columns of the database $\mathbf{T}$, i.e.,$\Delta(d, d') := \sum_{i=1}^{N} |t_i(d) - t_i(d')|$.*

The dimension distance is similar to the consecutive value difference for a single point, except that it is aggregated over all points in the database. If some of the dimensions have similar values and are correlated, then we expect their dimension distance to behave similarly to the differences of individual points and have a small value. If, however, they are uncorrelated, we expect their dimension distance to be much larger. Now we can rewrite the expression for $\bar{V}(\mathcal{D}, \mathcal{B})$,

$$\bar{V}(\mathcal{D}, \mathcal{B}) = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \notin \mathcal{B}}}^{D} |t_i(d_j) - t_i(d_{j-1})| = \sum_{\substack{j=2 \\ j \notin \mathcal{B}}}^{D} \Delta(d_j, d_{j-1}). \quad (1)$$

3.3 Partitioning with TSP

With multiple points, we can no longer find the optimal ordering and breakpoints via a simple sorting. However, as observed before, sorting the values in ascending (or descending) order is equivalent to finding the order that minimizes the envelope volume and we can still seek an optimum of $\bar{V}$. As explained in Definition 6, we expect the dimension distance to behave similarly to the individual differences; it should be small for dimensions with related values and large for uncorrelated dimensions.

Instead of optimizing simultaneously for $\mathcal{D}$ and $\mathcal{B}$, we will first optimize for $\mathcal{D}$ and subsequently choose the breakpoints in a fashion similar to Lemma 2. Therefore, our objective function $C(\mathcal{D})$ is similar to Equation (1), except that it also includes dimension distances across potential breakpoints,

Definition 7 (TSP objective). *We will optimize for the* cost objective

$$C(\mathcal{D}) := \sum_{j=2}^{D} \Delta(d_j, d_{j-1}), \tag{2}$$

This formulation implies that $\Delta(d_1, d_D) \geq \Delta(d_j, d_{j-1})$*, for* $2 \leq j \leq D$.

If the last condition were not true, a simple cyclical permutation of $\mathcal{D}$ would achieve a lower cost. After we have found $\mathcal{D}^* = \arg\max_{\mathcal{D}} C(\mathcal{D})$, we will select the breakpoints in a fashion similar to Lemma 2, by taking the indices of the top-(K–1) dimension distances $\Delta(d_j, d_{j-1})$, $2 \leq j \leq D$. This simplification of optimizing first for $\mathcal{D}$ has the added benefit that we can very quickly try different values of K. But the objective of Equation (2) is that of the *traveling salesman problem (TSP)*, where nodes correspond to dimensions and edge lengths correspond to dimension distances. In Figure 6(a) the TSP tour is shown with thick lines. The breakpoints (for $K = 2$) are its two longest edges (dashed thick lines). The sketch of the RIVA algorithm is given below:

1. Scan the database once to compute the $D \times D$ matrix of dimension distances.
2. Find a TSP tour $\mathcal{D}$ of the D dimensions, using the above distances.
3. If necessary, rotate it to satisfy the condition in Definition 7.
4. Choose the remaining $K - 1$ breakpoints in $\mathcal{B}$ as described above.

The column reordering problem for binary matrices, which is a special case of the desired reordering for our problem is already shown to be NP-hard [5]. Finally, the dimension distance Δ satisfies the triangle inequality, in which case a factor-2 optimal of $C(\mathcal{D})$ can be found in polynomial time. In practice, even better solutions can be found quite efficiently (e.g., for $D = 100$, typical running time for TSP using Concorde[1] is about 3 seconds).

4 Indexing

In the previous section we outlined how to find an ordered partitioning that makes the points as smooth as possible, with a single pass over the database. A natural low-dimensional representation of the points $\mathbf{t}_i$ is the per-partition average [9]. More precisely, we map each $\mathbf{t}_i \in \mathbb{R}^D$ into $\hat{\mathbf{t}}_i \in \mathbb{R}^K$ defined by

$$\hat{t}_i(k) := \tfrac{1}{D_k} \sum_{d \in \mathcal{D}_k} t_i(d), \quad \text{for } 1 \leq k \leq K.$$

Assume we want to index $\mathbf{t}_i$ using any L^p norm. For $1 \leq p \leq \infty$, we define the lower-bounding norm $\|\cdot\|_{lb(p)}$ on the low-dimensional representations $\hat{\mathbf{t}}_i$ as

$$\|\hat{\mathbf{t}}_i\|_{lb(p)} := \left(\sum_{k=1}^{K} D_k \cdot |\hat{t}_i(k)|^p\right)^{\frac{1}{p}}, \text{ if } p \neq \infty, \text{ or } \|\hat{\mathbf{t}}_i\|_{lb(\infty)} := \|\hat{\mathbf{t}}_i\|_{\infty}, \text{ if } p = \infty.$$

That $\|\cdot\|_{lb(p)}$ is a lower-bounding norm for the corresponding L^p norm on the original data $\mathbf{t}_i$ is a simple extension of theorems for equal-length partitions [9]. We index $\hat{\mathbf{t}}_i$ using an R-tree (see Figure 6(b) for a simple 2D example), which

[1] `http://www.tsp.gatech.edu/concorde/`

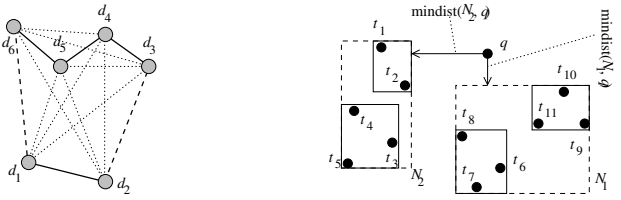


Fig. 6. (a) Illustration of TSP problem, (b) R-tree structure

recursively groups points into bounding boxes (nodes). A range query prunes nodes based on the minimum possible distance (*mindist*) of the query points to any point contained within a node. NN queries are processed by depth-first traversal and a priority queue, again using *mindist*. Since, $\|\hat{\mathbf{t}}_i\|_{lb(p)} \leq \|\mathbf{t}_i\|_p$, computing *mindist* using $\|\hat{\mathbf{t}}_i\|_{lb(p)}$ guarantees no false dismissals. We chose the partitioning $(\mathcal{D}, \mathcal{B})$ so as to make the segments as smooth as possible, therefore we expect both the node volumes to be small. Furthermore, it is precisely the smoothness that makes per-segment averages good summaries and $\|\hat{\mathbf{t}}_i\|_{lb(p)}$ a good approximation of $\|\mathbf{t}_i\|_p$.

5 Experiments

5.1 Example for Image Data

We depict with a running example the usefulness of the dimension reordering techniques for indexing and visualization. We utilize portions of the HHRECO[2] symbol recognition database, which consists of approximately 8000 shapes signed by 19 users. The user strokes are rendered on screen and treated as images (200×150). Since it would be unrealistic to treat each image as 200-by-150 dimensional point we perform a simple compaction of the image features as follows: by applying a $k \times m$ grid on the image, we record only $k \times m$ values which capture the number of pixels falling into each bucket. Using a 5×5 grid and starting from the top left image bucket we follow a *meander* ordering (Figure 7) and transform each image into a 25-dimensional point. The exact bucket ordering technique is of little importance, since the dimensions are going to be reordered again by our technique (therefore z- or diagonal ordering could be equally used).

On the right of Figure 7 we show the originally derived 25D points for 12 images of the dataset. Figure 8 depicts the new sequences after the TSP-based reordering and also the grouping of dimensions into 3 segments. Finally the same figure also illustrates the averaging per group of projected dimensions. The new projected dimensions correspond to a *group* of the original dimensions, the correspondence of which is shown in Figure 10. Image regions corresponding to empty image space are clustered together, while image portions that carry stroke information are grouped into different segments.

[2] http://www-cad.eecs.berkeley.edu/Respep/Research/hhreco/

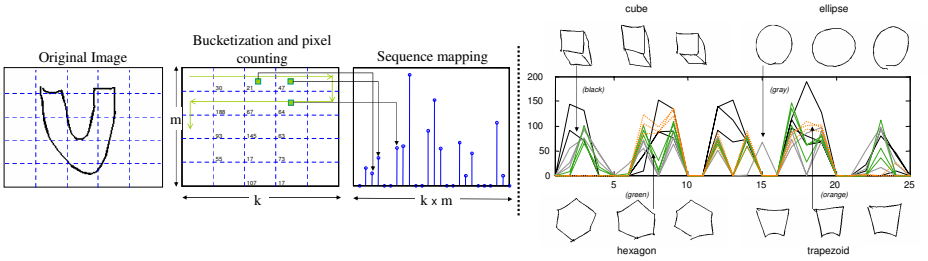


Fig. 7. *Left:* Feature extraction from image, *Right:* Mapping of features as sequences

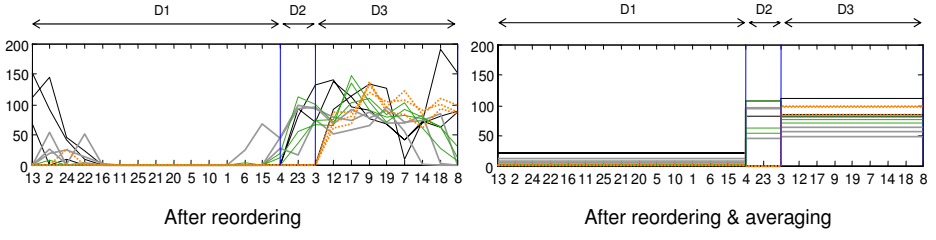


Fig. 8. *Left*: After dimension reordering, *Right*: After dimension averaging

Plots on projected dimensions (like Figure 8) can be very useful for summarizing and visualizing high-dimensional data. This mapping resembles the parallel coordinate technique [4]. However, our approach additionally *groups, reorders and summarizes* dimensions. When the images are projected into 2 or 3 groups of dimensions, they can also be visualized in 2D or 3D. For example, by projecting the 25D points onto 2D and placing the 12 images at their summarized projected coordinates we get the mapping of Figure 9. One can observe that

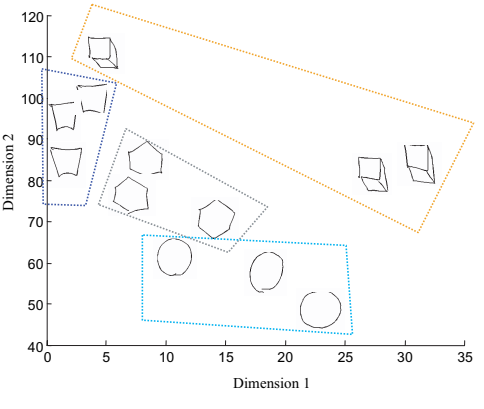


Fig. 9. 2D image mapping

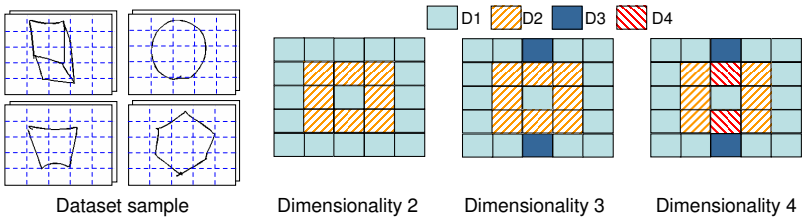


Fig. 10. Correspondence between projected dimensions and portions of the image for projected dimensionalities of 2, 3 and 4

relative distances are well preserved and similar-looking shapes (e.g., hexagons and circles) are projected in the vicinity of each other.

5.2 Application for Collaborative Filtering

We use the `MovieLens` database [3] which is utilized as a movie recommendation system. The database contains ratings for 1682 movies from 943 users. We sample a smaller portion of the database, containing all the ratings for 250 random movies and we apply our dimension ($\equiv$ movies) reordering technique. Indicative of the effective reordering is the measurement of the global smoothness, which is improved, since the cost function C that we are optimizing is minimized by a factor of 6.2. We also observed very meaningful groups of movies in the projected dimensions. For example, one grouping contains action blockbuster movies like "Indiana Jones", "Empire Strikes Back" and "Terminator", while another contains more action thrillers like "Conspiracy Theory" or "The Game".

5.3 Indexing with R-Trees

Now we quantify the performance gains of our reordering and dimension grouping techniques on indexing structures (and specifically on R-trees). For this experiment we use all the images of the `HHRECO` database, but we hold out 50 random images for querying purposes. Images are converted to high-dimensional points (as discussed before), using 9, 16, 36 and 64-dimensional features. These high-dimensional features are reduced down to 3, 4, 5, 6 and 8 dimensions using the proposed methodology. The original high-dimensional data are indexed in an R-tree and their low-dimensional counterparts are also indexed in R-trees using the modified *mindist* function as discussed in Section 4.

For each method we record the amount of retrieved high-dimensional data, i.e., how many leaf records are accessed. Figure 11 displays the results normalized by the total number of data. The R-tree on the original data exhibits very little pruning power which was expected, since it operates at high dimensionality. The new R-trees operating on the grouped dimensions exhibit much higher efficiency. Notice that for 9D original dimensionality we can improve the search

[3] http://www.grouplens.org/

performance by 78% in the best case, which happens for 6 grouped dimensions. For 16D data a projected group dimensionality of 8 is the one that gives the best results, which is 62% better than the pruning power of the original R-tree. For even higher data dimensionalities, the gain from the dimension grouping diminishes slowly but one should bear in mind that the original R-tree already fetches approximately all of the data for dimensionalities higher than 16. An interesting research direction for future work would be to examine the possibility of a connection between the projected group dimensionality at which the R-tree operates most efficiently and the intrinsic data dimensionality. Realization of such a connection can lead to more effective design of indexing techniques.

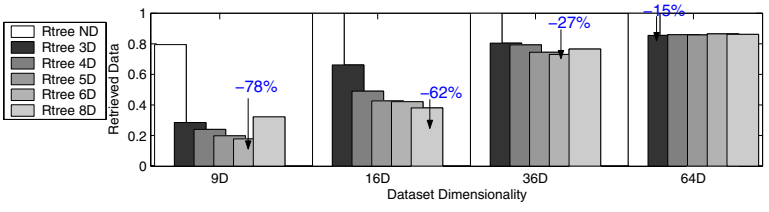


Fig. 11. Savings from using our projected grouping in conjunction with an R-tree. Data at various dimensionalities (x-axis) are projected down to 3, 4, 5, 6 and 8 dimensions.

Concluding, the indexing experiments have demonstrated that our methods can effectively enhance the pruning power of indexing techniques. We should also emphasize that essentially we have only *reorganized* and *packetized* differently the data dimensions, but we have not modified in the least in inner-workings or the structure of the R-tree index. Additionally, since there is a direct mapping between the grouped and original dimensions our technique has the additional benefit of enhanced interpretability of the results.

In the future, we plan to investigate additional ways of further improving the index performance on the new data representation. For example, in this work the grouped dimensions include all the original dimensions. However, some dimension groups are less important than others and therefore do not have to be indexed, leading to further reduction in the data dimensionality. One can see this as an analog of the largest principal components. Furthermore, such an addition to out methodology will allow for an indirect comparison with PCA, which is something that we also intend to explore in the immediate future.

6 Related Work

Traditional clustering approaches, such as K-means, K-medoids or hierarchical clustering focus on finding groups of similar values and not on finding a smooth ordering, which is the main target of this work. In the mutually related fields of co-clustering, bi-clustering, subspace clustering [1,8] (for a detailed review, see [7]) and graph partitioning [6], the problem of finding pattern similarities

has been explored. Among those, pCluster [8] also tries to minimize pairwise differences, both among dimensions as well as among tuples. In general, all of these approaches focus on clustering both rows and columns and treat them symmetrically. In contrast, we assume an asymmetry ($N \gg D$) which makes the solution quite different. Most of these approaches are not suitable for large-scale databases with millions of tuples.

Mamoulis et al. [2] propose a vertical partitioning scheme for nearest neighbor query processing, which considers columns in order of decreasing variance. As pointed out before, our cost objective is not related to the per-column variance. More importantly, [2] does not provide any grouping of the dimensions, and hence is not suitable for visualization or indexing. Our dimension summarization technique bares resemblances to the *piecewice aggregate approximation (PAA)* and *segment means* [9]; however our scheme is more general, in that, it allows segments of unequal size. Additionally those techniques perform no reordering, since they are predicated on the smoothness assumption of time-series data.

In the area of high-dimensional visualization, Fastmap [3] is a popular and fast method for dimensionality reduction and visualization. Nonetheless, it does not provide any bounds on the distance in the low-dimensional space, and therefore cannot guarantee the *no false dismissals* claim, that is provided by our lower-bounding criterion. Finally, our data representation can be seen as an extension of the *parallel coordinates* method [4]. Our technique enriches the previous model, making visualizations more coherent and useful, not only because it provides a much smoother representation, but because it also performs the additional steps of dimension grouping and summarization.

7 Conclusion

We presented RIVA, a new methodology for indexing and visualizing high-dimensional data. By expressing the data in a parallel coordinate system we attempt to discover a dimension ordering that will provide a globally smooth data representation. Such a data representation is expected to minimize data overlap and therefore enhance generic index performance as well as data visualization. We solve the dimension reordering problem by recasting it as an instance of the well-studied TSP problem. Our results indicate that R-tree performance can reap significant benefits from this dimension reorganization.

References

[1] C. C. Aggarwal, J. Han, J. Yang, and P. S. Yu. A framework for projected clustering of high dimensional data streams. In *VLDB*, 2004.

[2] A. P. de Vries, N. Mamoulis, N. Nes, and M. L. Kersten. Efficient k-NN search on vertically decomposed data. In *SIGMOD*, 2002.

[3] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD*, 1995.

[4] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multidimensional geometry. In *IEEE Visualization '90*, 1990.
[5] D. Johnson, S. Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *VLDB*, 2004.
[6] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *SC98*, 1998.
[7] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Trans. Comp. Biol. and Bioinf.*, 1(1), 2004.
[8] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *SIGMOD*, 2002.
[9] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary $\mathcal{L}_p$ norms. In *VLDB*, 2000.

Distributed Subgroup Mining

Michael Wurst and Martin Scholz

Artificial Intelligence Group, University of Dortmund, Germany
{wurst, scholz}@ls8.cs.uni-dortmund

Abstract. Subgroup discovery is a popular form of supervised rule learning, applicable to descriptive and predictive tasks. In this work we study two natural extensions of classical subgroup discovery to distributed settings. In the first variant the goal is to efficiently identify global subgroups, i.e. the rules an analysis would yield after collecting all the data at a single central database. In contrast, the second considered variant takes the locality of data explicitly into account. The aim is to find patterns that point out major differences between individual databases with respect to a specific property of interest (target attribute). We point out substantial differences between these novel learning problems and other kinds of distributed data mining tasks. These differences motivate new search and communication strategies, aiming at a minimization of computation time and communication costs. We present and empirically evaluate new algorithms for both considered variants.

1 Introduction

The aim of data mining is to find useful patterns in large data collections. Distributed computing plays an important role in this process for several reasons. First, data mining often requires huge amounts of resources in storage space and computation time. To make systems scalable, it is important to develop mechanisms that distribute the work load among several sites in a flexible way. Second, data is often inherently distributed to several databases, making a centralized processing of this data very inefficient and prone to security risks. Algorithms for several data mining tasks were proposed, for example for distributed association rule mining [1], classification, clustering and dimensionality reduction [2].

The goal of subgroup discovery [3] is to identify interesting rules. In this paper we study two distributed subgroup discovery tasks. The first one aims at the discovery of global subgroups from distributed databases using distributed algorithms. Its two objectives are to find the same rules as centralized learners [4], and to minimize communication costs. The second distributed task aims to detect relative local subgroups, which describe characteristics of individual databases concerning a target value. Such rules can, for instance, capture information how sales deviate at certain branches of a company.

In Sec. 2 existing work on subgroup discovery is presented and extended to distributed settings in Sec. 3. Novel algorithmic solutions for distributed subgroup discovery are proposed in Sec. 4 and evaluated in Sec. 5.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 421–433, 2006.
© Springer-Verlag Berlin Heidelberg 2006

2 Subgroup Discovery

Subgroup discovery aims at the identification of interesting and interpretable rules [3,5]. Each subgroup describes a subset of the overall population in a database, which deviates from the overall behavior in terms of a given property of interest. Among the typical applications of subgroup discovery are the identification of homogeneous groups of clients in marketing, and the induction of interpretable rules in medical domains. Case studies illustrating benefits of subgroup discovery for decision support can be found in [6]. An example application in a medical domain with a focus on incorporating background knowledge and user-defined preferences is described in [7].

Formally, the learning task shares some of the characteristics of classifier induction. In particular it is also a supervised learning task. Instances of the population are referred to as *examples* $x \in \mathcal{X}$ in this paper, while the property of interest can be formalized as a target attribute $\mathcal{Y}$. Every subgroup is represented by a rule $A \rightarrow C$ with antecedent $A : \mathcal{X} \rightarrow \{true, false\}$ that is a conjunction of literals, and a conclusion $C : \mathcal{Y} \rightarrow \{true, false\}$.

In subgroup discovery the interestingness of rules is measured in terms of a user-given *utility function*, a parameter of the task itself. Hence, subgroup discovery can be considered to define a very broad rule discovery framework, covering classification as a specific case. However, the focus of subgroup discovery differs from inducing classifiers. Although the discovered sets of rules can well be used to make predictions if interpreted probabilistically [8], subgroup discovery is typically used for descriptive data analysis. Different selection metrics reflecting rule interestingness have been motivated in [5]. The main objective of utility functions is to trade-off between two quantities, which both indicate interestingness but tend to be diametric. These quantities are formalized in the following two definitions. To ease notation we denote the absolute number of true positives of a rule r with $p(r)$, and the number of its false positives with $n(r)$. The argument is omitted if clear from the context. P and N denote the number of positives and negatives in the complete dataset.

Definition 1. *For a given database* $E \subseteq X \times Y$ *the* support *of a rule* $A \rightarrow C$ *is denoted as* $Sup(A \rightarrow C)$*. It is defined as*

$$Sup(A \rightarrow C) := \frac{|\{A(x) \mid \langle x, y\rangle \in E\}|}{|E|} = p + n,$$

the fraction of examples $\langle x, y\rangle \in E$ *for which the antecedent* A *evaluates to true.*

The notion of rule support is well-known from frequent itemset mining [9].

Definition 2. *The* bias *of a rule* $r : A \rightarrow C$ *is defined as the difference between the conditional distribution of* C *given* A *and the default probability of* C*:*

$$Bias(r) := \frac{p}{p+n} - \frac{P}{P+N}.$$

The bias reflects the degree to which a subgroup differs from expectation, i.e. that the target attribute is distributed as in the overall dataset. A broad variety of utility functions have been suggested for rule discovery, most of which are monotone in the bias and support of rules. Please refer to [10] for an overview. The most popular utility function for subgroup discovery is the weighted relative accuracy [5,11], which is exemplarily used in our algorithms proposed in Sec. 4.

Definition 3. *The* weighted relative accuracy *of a rule r is defined as*

$$WRAcc(r) := Sup(r) \cdot Bias(r) = \frac{p+n}{P+N}\left(\frac{p}{p+n} - \frac{P}{P+N}\right).$$

For selecting rules only the order induced by an evaluation metric is relevant. Since P and N are constants for any fixed dataset we may multiply the term above with $\frac{(P+N)^2}{N}$ and reach at a more convenient formulation:

$$\begin{aligned} WRAcc \cdot \frac{(P+N)^2}{N} &= (p+n) \cdot \frac{(P+N)}{N} \cdot \left(\frac{p}{p+n} - \frac{P}{P+N}\right) \\ &= \frac{p \cdot (P+N) - (p+n) \cdot P}{N} = p - \frac{P}{N} \cdot n = p - c \cdot n \end{aligned} \tag{1}$$

for a database-dependent constant $c \in \mathbb{R}^+$. This simple reformulation reflects the fact that ROC isometrics of the weighted relative accuracy are parallel lines with a slope of 1. Learners optimizing this evaluation metric handle class skews differently than e.g. predictive accuracy does. Several search strategies have been proposed to find rules optimizing this metric for different settings.

The most straight-forward subgroup discovery task is to identifying a set of k rules with highest utility scores, where k is a user-given parameter. The ILP system MIDOS [4] is the best-known algorithm for this task, optimizing the weighted relative accuracy metric. It is designed for multi-relational learning and searches the space of rules exhaustively, except for safe pruning. The use of a refinement operator allows to evaluate rules from general to specific, while making sure that no rule is evaluated twice. The pruning strategy exploits the operator's top-down search. The support of each rule r decreases monotonically with each refinement, so for p positive and n negative examples the upper bound

$$WRAcc(r) \leq Sup(r) \cdot \left(1 - \frac{P}{P+N}\right) \tag{2}$$

allows to prune all refinements of a rule with low support if it cannot improve over the k-th best rule found so far.

In order to speed up the subgroup discovery process adaptive sampling has been proposed [12]. The learning task needs to be reformulated to account for the inevitable risk of drawing a poor sample. Hence, the goal is to find k approximately best rules with high probability. For the most relevant utility functions probabilistic guarantees can be given to find good rules with high probability.

Several authors have addressed subgroup discovery in the presence of (or relative to) prior knowledge. A recently presented system exploits different kinds of background knowledge to select relevant features, to discretize variables in a meaningful way, and to exploit user-given preferences for guiding search heuristics [7]. Knowledge-based sampling [8] generically incorporates probabilistic prior knowledge and can be combined with any of the other approaches. It yields unexpected patterns and supports the induction of accurate classifier ensembles.

3 Discovering Subgroups from Distributed Data

3.1 Global Distributed Subgroup Mining

An extension to classical subgroup discovery that has not yet been investigated by the data mining community is the discovery of subgroups from distributed data. We start with a few definitions for evaluating rules on distributed data to ease the formulation of the formal learning problems studied in this work.

In the remainder of this paper the global data E is assumed to be distributed to nodes $\{1, \ldots, m\}$, each holding a local subset $E_i \subset E$ so that $E = \biguplus_{i=1}^{m} E_i$. The number of positives and negatives at site i are denoted as P_i and N_i.

Definition 4. *For any rule $r : A \rightarrow C$ the absolute number of covered positives and covered negatives in node i are denoted as*

$$p_i(r) := |\{A(x) \wedge C(y) \mid \langle x, y \rangle \in E_i\}| \quad \textit{and} \quad n_i(r) := |\{A(x) \wedge \overline{C(y)} \mid \langle x, y \rangle \in E_i\}|.$$

This allows to restate the support and bias of rules for individual databases E_i.

Definition 5. *The* local support *of rule r at a site i is defined as*

$$Sup_i(r) := \frac{p_i(r) + n_i(r)}{|E_i|},$$

while the local bias *is defined as*

$$Bias_i(r) := \frac{p_i(r)}{p_i(r) + n_i(r)} - \frac{P_i(r)}{P_i(r) + N_i(r)}$$

Global utility functions can be adapted in a straight-forward manner based on these local quantities. We confine ourselves to weighted relative accuracy.

Definition 6. *Local weighted relative accuracy of rule r at node i is defined as*

$$WRAcc_i(r) := Sup_i(r) \cdot Bias_i(r).$$

The first studied distributed subgroup discovery task is referred to as *global subgroup discovery*. It aims at the identification of the same k best subgroups in the global data E, but without shifting all the data to a single database.

Global subgroup discovery is an unexpectedly hard problem. If the distribution underlying different databases E_i may deviate from the global distributions,

i.e. they cannot be considered to be uniform subsamples of E, then globally best rules may perform poor at *all* local sites [13]. More precisely, collecting all the locally best rules with respect to $WRAcc_i$ does not necessarily yield a set that contains one of the k globally best rules, neither exactly nor approximately in the sense of the approximately k best rules problem (see Sec. 2). As a consequence, algorithms addressing global subgroup discovery need to exchange either examples or models and counts if guarantees are required. A new algorithm tailored towards the specific characteristics of the task will be presented in Sec. 4.1.

3.2 Relative Local Subgroup Discovery

The novel task of relative subgroup mining takes the locality of data explicitly into account. A rule is considered to be interesting, if it is well supported by local data, and if its local confidence deviates substantially from the corresponding confidence when evaluating the same rule globally.

Relative subgroups are relevant in several domains. E.g. in a marketing application the corresponding rules may identify spatial regions in which the buying behavior of customers differs from that observed in other parts of the country. An unsupervised approach with a related aim, mining *high contrast frequent itemsets*, has recently been presented [14]. Based on entropy, it identifies itemsets with counts that are inhomogeneously distributed to the different sites. In this paper we address supervised relative rule discovery, a learning task proposed in recent prior work [13]. It aims at the identification of rules maximizing the following evaluation metric:

Definition 7. *The* relative local utility *of a rule r at node i is defined as*

$$RLU_i(r) := Sup_i(r) \cdot (Bias_i(r) - Bias(r) + c_i)\,, \text{ with } \; c_i := \frac{P_i}{P_i + N_i} - \frac{P}{P + N}.$$

Different class skews P_i/N_i are of minor interest in this setting, so the term c_i is used to focus on deviations of globally and locally different conditional class distributions for subsets covered by considered rules. This turns the term in brackets into deviations of local and global confidences, as motivated above.
As for $WRAcc$, a more convenient version of the RLU metric can be derived:

$$\begin{aligned} RLU_i(r) &= Sup_i(r) \cdot \left(\frac{p_i(r)}{p_i(r) + n_i(r)} - \frac{p(r)}{p(r) + n(r)} \right) \\ &= |E_i|^{-1} \cdot \left(p_i(r) - \underbrace{p(r) \cdot \frac{p_i(r) + n_i(r)}{p(r) + n(r)}}_{=:\hat{p}_i(r)} \right) = \frac{p_i(r) - \hat{p}_i(r)}{|E_i|} \end{aligned}$$

The term $\hat{p}_i(r)$ can be interpreted as the estimated number of positives within the subset covered by rule r at site i. This estimate is based on the fraction of positives in the subset of the *global* data that are covered by the rule, i.e. on the global confidence. A factor-equivalent metric to RLU is $RLU_i^*(r) := p_i(r) - \hat{p}_i(r)$.

The task of discovering the best k relative local subgroups has been shown to be at least as hard as discovering global subgroups from distributed data [13].

4 Algorithms for Distributed Subgroup Discovery

4.1 Distributed Global Subgroup Discovery

In this section we propose an algorithm for distributed global subgroup mining based on count polling and distributed rule pruning based on optimistic estimates. A basic principle of the algorithm is that for each rule r all refinements of this rule r' are created and counted at exactly one node. We use a refinement operator as defined in [4]. The following definition assumes a fixed total order on the set of attributes.

Definition 8. *A refinement operator ρ is a function that maps each rule to the set of its direct successors. A rule $r' : A' \rightarrow C'$ is a direct extension of $r : A \rightarrow C$, if and only if $C = C'$, $A' = A \cup \{X_i = v\}$ for a variable X_i with the property that all attributes X_j in A have an index j which is strictly lower than i. The transitive relation $r' < r$ denotes, that r' is a refinement of r.*

Our pruning method exploits the following relationship. If for each node the counts for a rule r or a predecessor of r, denoted as r' are known, we can calculate a tight upper bound on the $WRAcc(r)$. If this highest possible score is worse than the currently k-best rule, then the algorithm can safely prune the rule r.

Lemma 1. *The (global) utility of a rule r is bounded by the following term*

$$WRAcc(r) \leq \frac{\sum_{i=1}^{m} p_i(r'_i)}{P+N} \cdot \left(1 - \frac{P}{P+N}\right) = \frac{N}{(P+N)^2} \sum_{i=1}^{m} p_i(r'_i),$$

where $r'_i = r$ or $r < r'_i$. For the most specific rules $p_i(r_i)$ is known for, this bound is tight.

Proof. The correctness of the lemma follows from eqn. (1), implying that $WRAcc$ is order-equivalent to $p - \frac{P}{N} \cdot n$. Hence, optimal refinements discard all negatives but no positives, which leads to the score used as an upper bound.

The difference to eqn. (2) is that the support is replaced by the fraction of true positives, a quantity which is strictly smaller unless r cannot further be improved by refinements, anyway. The pruning strategy exploits the fact that $WRAcc$ increases monotonically if refinements "discard" only negatives. It is maximized by refinements that discard all negatives and no positives. For this reason straightforward adaptations of eqn. (2) apply to the broad class of utility functions sharing this property of monotonicity, e.g. to the binomial test function. It is sufficient to substitute the tightest known counts during optimistic

score computation in lemma 1 for each rule, and to optimistically assume that a subsequent refinement is able to discard only the covered negatives.

The lemma can be used to prune rules for which exact counts are available only from a subset of all nodes. If the upper bound for $WRAcc(r)$ is worse than the k-th best rule, r can be pruned without polling further counts. Lemma 1 also implies a second pruning condition. If a rule r' is pruned, then all refinements $r < r'$ of this rule can be pruned as well, as their optimistic scores are known to be no better than the optimistic score of r'.

These pruning strategies are combined with count polling to derive an algorithm for distributed subgroup mining that scales linearly in the number of nodes. Each node i keeps three data structures. First, a list B_i containing the k currently best hypotheses. Second, a list of pruned hypotheses Z_i. These are rules for which it is known that no descendant can reach a score better than

$$kb_i := \min_{r \in B_i} WRAcc(r),$$

the k-th best score at node i. To this end an optimistic upper-bound is computed using lemma 1. Finally, each node keeps a list of all rules, for which it is polling counts. This list is denoted as Q_i.

The algorithm is initialized by assigning all rules with an empty body to an arbitrary node. The computation then follows Fig. 1. A node that receives an assignment for a rule r generates all canonical refinements (direct successors) $\rho(r)$ and serves as their polling node. A rule $r : A \to C$ can be pruned (i) based on its optimistic score, or (ii) because it is *subsumed* by a previously pruned rule $r' : A' \to C'$, that is $C' = C$ and $A' \subset A$, so $\{A(x) \mid x \in \mathcal{X}\} \subset \{A'(x) \mid x \in \mathcal{X}\}$ and hence $p_i(r) < p_i(r')$ at all sites. For each refined rule r' the algorithm first obtains the local counts from the database and checks whether r' can be pruned. If the rule is pruned based on its optimistic score, the node additionally informs all other nodes about this step of pruning. In contrast, subsumption-based pruning of a rule r' does not require to broadcast r', since each node is known to have a rule subsuming r' in its list of pruned rules Z_i. If a rule is not pruned the node broadcasts a query for counts on r and adds r to the list of open hypotheses Q_i. The individual nodes then reply their local counts for r'. As more and more local counts arrive the bound on the global count gets tighter.

If all local counts for a rule r are available and r cannot be pruned, it is first checked, if the rule is better than kb_i. If this is the case, it is inserted into B_i as described above and broadcasted to all other nodes. Then the rule is assigned to a node that is responsible for generating and counting the canonical refinements of the rule. Besides the rule itself, the local counts for rule r are transmitted from all the nodes. This information is necessary to allow for pruning based on partially available counts, as described above. The node to which a rule is assigned is determined by the support of the rule. The rationale of this choice is, that such a node is the most likely to be able to prune the rule without querying other nodes for counts.

The algorithm has communication costs in $O(m|C|)$, where m is the number of nodes and C is the set of evaluated candidates. Hence, the algorithm scales

```
// Update best rules
for best_ij(r, WRAcc(r)) ∈ M_j do
  if WRAcc(r) > kb_j then
    insert(B_j,r);

// Update pruned rules
for prune_ij(r) ∈ M_j do
  Z_j = Z_j ∪ {r};

// Obtain message counts
for count_ij(r, n_i(r), p_i(r)) ∈ M_j do
  recalculate optscore(r);
  if prunable(r) then
    Z_j = Z_j ∪ {r};
  else
    if counts-complete(r) then
      if WRAcc(r) > kb_j then
        best.insert(B_j, r);
        bcast(best(r, WRAcc(r)));
      Q_j = Q_j \ {r};
      m = argmax_i(n_i(r) + p_i(r));
      send(assign_jm(r, {(p_1(r), ...)}));

// Handle assignment to refine a rule
for assign_ij(r, {(p_1(r), ...)}) ∈ M_j do
  for r' ∈ refinements(r) do
    recalculate optscore(r');
    if not(prunable(r')) then
      bcast(query(r'));
      Q_j = Q_j ∪ {r'};

// Answer queries for local counts
for query_ij(r) ∈ M_j do
  send(count_ji(r, n_j(r), p_j(r)));

prunable(r):
  if r ≤ r' : r' ∈ Z_j then
    return true;
  if optscore(r) < kb_j then
    bcast(prune(r));
    return true;
  return false;
```

Fig. 1. Algorithm for distributed global subgroup mining at node j. M_j denotes the input message queue of node j. $best_{ij}$, $prune_{ij}$, $count_{ij}$, $query_{ij}$ and $assign_{ij}$ are messages, where i denotes the sender and j the receiver. The procedures above are executed as long as messages arrive.

linearly with the number of nodes and candidates. This can easily be seen from the fact that at most $O(m)$ messages are exchanged per candidate: a query for counts, its replies, and possibly a broadcast for a new best hypothesis *or* for pruning. These messages contain only rules and individual counts. Additionally, at most one delegation message for a rule is produced, containing a set of local counts. This message is of size $O(m)$.

4.2 Distributed Relative Subgroup Discovery

Finding relative local subgroups differs from finding global subgroups in that each node finds an own, individual set of rules. The score of a rule is defined with respect to its local support and its relative bias. While the support of a rule r can easily be computed locally at each database, global counts for r are required for computing the bias. Global counts of rules are aggregated as described in the last section. There is one important difference however. Rules can only be pruned, if they are pruned at *every* node. We propose an algorithm that is based on count polling and optimistic pruning. The following tight optimistic pruning rule holds for the task of relative local subgroup mining.

Lemma 2. *For relative local subgroup discovery, rules r with $p_i(r)$ positives, $n_i(r)$ negatives, and $\hat{p}_i(r)$ estimated positives covered by rule r at site i,*

$$RLU_i(r') \leq \frac{p_i(r) - \max(0, \hat{p}_i(r) - n_i(r))}{|E_i|}$$

is a tight upper-bound for the local utilities of all rules $r' < r$.

Proof. Considering the factor-equivalent metric RLU^* it is easily seen that an optimal refinement of rule r reduces $\hat{p}_i(r)$ by covering less examples that are "predicted" positive, while not reducing $p_i(r)$. If the $n_i(r)$ negative examples covered by r are predicted positive by $\hat{p}_i(r)$, and if a refinement $r' < r$ exists that covers only the $p_i(r)$ positive examples, then we reach at a utility of

$$RLU_i^*(r') = p_i(r) - \max(0, \hat{p}_i(r) - n_i(r)).$$

This cannot be improved any further by refinements, since r' covers only positives, and further refinement reduces $p_i(r)$ at least as much as $\hat{p}_i(r) - n_i(r)$. Since $RLU^* = RLU \cdot |E_i|$ this proves the lemma.

Our algorithm for relative subgroup mining works as follows. Again, each node has a list of best rules, pruned rules, and open rules. Additionally, nodes keep a rule cache, that is used to store the global counts of rules for which a node serves as the polling node. The mapping of rules to responsible nodes is realized by a hash function.

Each node starts with an empty set of rule candidates. It then generates first-level rules that are evaluated locally. If a rule r can be pruned based on lemma 2 it is discarded. Otherwise, the node requests global counts $n(r)$ and $p(r)$ for r from a polling node that is determined by calculating a hash value for the rule. The node that receives this request checks whether it finds the rule in its cache. If so, it directly returns the corresponding global counts. Otherwise, the node first queries all other nodes for their corresponding local counts. After aggregating all local counts $n_i(r)$ and $p_i(r)$ the polling node stores and returns the global counts. Given the global counts and the local counts for a rule r, the exact utility score of r can be computed. If r is better than the k-best rule it is inserted into B_i as described in the last section. If r, and thus each of its refinements, receive an optimistic score that is worse than the lowest score in B_i, then r is pruned. Neither best rules nor pruned rules are broadcasted, as they are not relevant to other nodes.

While the pruning strategies for relative local subgroup mining are weaker than for distributed global subgroup mining, the approach still scales linearly with the number of nodes. Thus, relative local subgroup mining is in $O(|C|m)$, where $|C|$ are the candidates considered by at least one node. Relative local subgroup mining for all nodes is usually more expensive than global subgroup mining, because rules may only be pruned, if they would be pruned at all nodes.

5 Experiments

We performed experiments to analyze the properties of the proposed algorithms. As both algorithms are guaranteed to find the best rules, evaluation is only concerned with communication costs. These costs are evaluated on three datasets taken from the UCI library, mushroom, adult, and german. For adult and german numerical attributes were discretized using minimal entropy partitioning.

First of all the substantial difference between the tasks of subgroup and association rule mining is illustrated exemplarily. Association rule and frequent itemset mining rely on a user-provided support threshold and are usually applied to find huge amounts of rules. Subgroup discovery finds only the k best rules with respect to a user-specified utility function, not requiring a threshold. Even if the best rule utility was known to a frequent itemset mining algorithm in advance, it would be more costly to generate all itemsets based on a corresponding support threshold in a distributed setting than to run distributed subgroup discovery; state-of-the-art algorithms for distributed frequent itemset mining evaluate at least all frequent itemsets at all nodes. E.g. the german dataset contains more than 50.000 frequent itemsets using the support-based pruning threshold of the MIDOS algorithm (see eqn. (2)) in combination with the (usually unknown) utility of the best subgroup. In contrast, the global subgroup discovery algorithm evaluates less than 3.000 *candidates*.

Still, the communication costs for our algorithm grow no more than linearly in the number of nodes. We validated this property in a first experiment, measuring costs by accounting 4 bytes for each rule transmitted over the network and 2 bytes for each count. To be able to measure the impact of data skews in the distribution of data to individual nodes we used the following procedure. First, the data was clustered using an EM algorithm. The number of clusters was chosen as the number of nodes. We use a parameter p_{skew} denoting the probability that an example is assigned to a node according to the corresponding cluster. Otherwise it is assigned randomly at equal probability. For $p_{skew} = 1$ each node receives all data points in its corresponding cluster. For $p_{skew} = 0$ all examples are distributed randomly. This allows to adjust the data skew between both extremes. The results for the datasets using $p_{skew} = 0$ and finding one global rule ($k = 1$) for rules of constrained length as in MIDOS (we searched for best rules containing up to 3 literals) are shown in Fig. 2. For all three datasets the curves confirm our theoretical findings concerning the scalability of our method. Please note, that in this experiment each database contains about the same amount of data, which is the worst case for our method.

The second experiment compares the communication costs for distributed global and relative subgroup mining for varying degrees of skew. The results of mining the most interesting rule of length up to 3 literals for the mushroom data set is shown in Fig. 3 for a network of $m = 5$ nodes. We see that distributed global subgroup mining shows a low sensitivity regarding the data skew. For relative subgroup mining the situation is different. Given a low skew, the costs for finding relative subgroups increases. The reason is that relative subgroups can only be found if the data distribution among nodes deviates. For low skews only

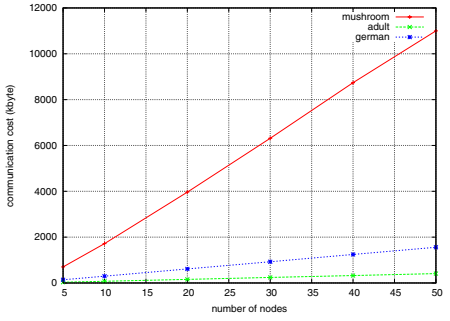


Fig. 2. Communication costs for distributed global subgroup mining

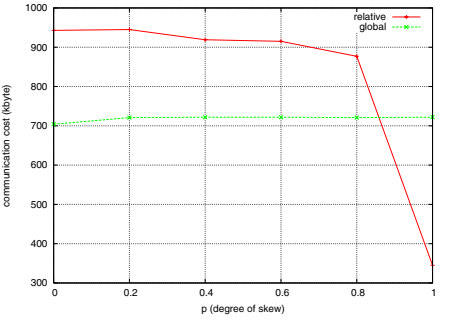


Fig. 3. Data skew / communication costs for global and relative subgroup mining

rules with very low scores can be identified, which however forces all nodes to search a very large search space as pruning cannot be applied. Reaching at a certain level of skew the distributions deviate sufficiently to identify corresponding logical rules, leading to a sharp decrease of costs in Fig. 3 for relative subgroup mining.

6 Discussion and Conclusion

Discovering distributed global and relative local subgroups are two novel knowledge discovery tasks. Since subgroup discovery is a supervised learning task it could be approached with state of the art distributed classification algorithms, e.g. distributed boosting [15] in order to find probabilistic rule ensembles as in [8]. Distributed boosting and similar algorithms are however not complete, thus do not guarantee to find optimal rules. As noted in [15] the quality of rules that can be discovered depends on the distribution of examples over the individual databases. Results presented in [13] support this observation. For this reason we focused on complete algorithms for distributed rule mining.

Existing complete algorithms for distributed rule mining are mostly concerned with finding association rules [1]. A straightforward extension of the Apriori algorithm is Count Distribution (CD) [16]. At each round, every database generates all $k+1$ candidates from the globally large k-itemsets and broadcasts all counts to all other nodes. This procedure causes communication costs of $\Omega(|C|m^2)$, where $|C|$ is the number of candidates and m is the number of nodes. One way to improve the CD algorithm is to use a designated node for each candidate that is responsible for polling and redistributing all counts of the candidate itemset. This method is applied in the FDM algorithm [17]. It reduces the communication complexity of the algorithm to $\Theta(|C|m)$. Two additional pruning techniques are applied in FDM. Local pruning is based on the observation that for an item to be frequent it must be frequent at least at one node. Only for such items counts need to be exchanged. Second, nodes use an optimistic estimate for the support of an itemset based on partial counts received from other nodes. If this estimate

is smaller than the minimal support, the candidate can be pruned. The idea of a polling site, as introduced by FDM helps to avoid costly broadcasts and is very general.

The real power of the above approaches lies in their local pruning strategies, however, which do not apply to distributed global subgroup mining as shown in [13]; globally optimal rules can simultaneously be inferior at each individual node, while pruning strategies applied to distributed frequent itemset mining rely on the fact that globally frequent itemsets *must* be frequent at least at one node. This reflects that subgroup utility functions are lacking the monotonicity of rule support, a prerequisite for efficient itemset mining. This substantial difference remains even for more sophisticated pruning strategies as proposed in [14,18].

Association rule based approaches are not applicable to relative subgroup mining either, because the relative score of each rule does not only depend on its local support, but also on the (independent) local and global rule confidences.

Hence, we presented two new algorithms for distributed subgroup discovery that guarantee to deliver optimal rules at communication costs linear in the number of nodes and rule candidates, an essential property for scalable distributed algorithms. The complexity was shown theoretically and confirmed empirically.

References

1. Zaki, M.J.: Parallel and Distributed Association Mining: A Survey. IEEE Concurrency **7** (1999)
2. Park, B.H., Kargupta, H.: Distributed Data Mining: Algorithms, Systems, and Applications. In Ye, N., ed.: Data Mining Handbook. IEA (2002)
3. Klösgen, W.: Subgroup discovery. In: Handbook of Data Mining and Knowledge Discovery. Oxford University Press (2002)
4. Wrobel, S.: An Algorithm for Multi–relational Discovery of Subgroups. In: Principles of Data Mining and Knowledge Discovery: First European Symposium (1997)
5. Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In Advances in Knowledge Discovery and Data Mining. AAAI Press (1996)
6. Lavrac, N., Cestnik, B., Gamberger, D., Flach, P.: Decision support through subgroup discovery: three case studies and the lessons learned. MLJ **57** (2004)
7. Atzmüller, M., Puppe, F., Buscher, H.P.: Exploiting background knowledge for knowledge-intensive subgroup discovery. In: Proc. of IJCAI (2005)
8. Scholz, M.: Sampling-Based Sequential Subgroup Mining. In: Proc. of KDD (2005)
9. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large data bases. In: Proc. of VLDB (1994)
10. Fürnkranz, J., Flach, P.: ROC 'n' Rule Learning – Towards a Better Understanding of Covering Algorithms. MLJ **58** (2005)
11. Nada Lavrac, N., Flach, P., Zupan, B.: Rule Evaluation Measures: A Unifying View. In: 9th International Workshop on Inductive Logic Programming (1999)
12. Scheffer, T., Wrobel, S.: Finding the Most Interesting Patterns in a Database Quickly by Using Sequential Sampling. JMLR **3** (2002)
13. Scholz, M.: On the Tractability of Rule Discovery from Distributed Data. In: Proc. of ICDM (2005)

14. Otey, M.E., Parthasarathy, S., Wang, C., Veloso, A., Meira, W.: Parallel and Distributed Methods for Incremental Frequent Itemset Mining. IEEE Transactions on Systems, Man, and Cybernetics, Part B **34** (2004) 2439–2450
15. Lazarevic, A., Obradovic, Z.: Boosting algorithms for parallel and distributed learning. Distributed and Parallel Databases Journal **11** (2002)
16. Agrawal, R., Shafer, J.C.: Parallel mining of association rules. IEEE Transactions On Knowledge And Data Engineering **8** (1996)
17. Cheung, D., Han, J., Ng, V., Fu, A., Fu, Y.: A Fast Distributed Algorithm for Mining Association Rules. In: International Conference on Parallel and Distributed Information Systems. (1996)
18. Schuster, A., Wolff, R.: Communication-efficient distributed mining of association rules. In: Proc. of SIGMOD (2001)

Network Flow for Collaborative Ranking

Ziming Zhuang[1], Silviu Cucerzan[2], and C. Lee Giles[1]

[1] Information Sciences and Technology, The Pennsylvania State University, University Park, PA 16802, U.S.A
[2] Microsoft Research, One Microsoft Way Redmond, WA 98052, U.S.A
{zzhuang, giles}@ist.psu.edu, silviu@microsoft.com

Abstract. In query based Web search, a significant percentage of user queries are underspecified, most likely by naive users. Collaborative ranking helps the naive user by exploiting the collective expertise. We present a novel algorithmic model inspired by the network flow theory, which constructs a search network based on search engine logs to describe the relationship between the relevant entities in search: queries, documents, and users. This formal model permits the theoretical investigation of the nature of collaborative ranking in more concrete terms, and the learning of the dependence relations among the different entities. *FlowRank*, an algorithm derived from this model through an analysis of empirical usage patterns, is implemented and evaluated. We empirically show its potential in experiments involving real-world user relevance ratings and a random sample of 1,334 documents and 100 queries from a popular document search engine. Definite improvements over two baseline ranking algorithms for approximately 47% of the queries are reported.

Keywords: Graph models, Network flow, Graph theory, Collaborative ranking, User feedback, FlowRank.

1 Introduction

The intuition behind collaborative ranking is that in the context of Web search collective knowledge enhances the ranking of search results. A number of methods to facilitate collaborative ranking have been proposed and some effectively implemented, particularly in online recommendation systems. By introducing a formal model, this work systematically investigates the nature of collaborative ranking.

Why does collaborative ranking work? Given the context of relevant queries, naïve queries can be improved by looking at relevant and articulated queries (and corresponding search results) from other users. By observing the collective patterns of access, the ranking system learns to favor certain search results. For the most part, existing collaborative ranking algorithms take into account the intricate relationships between relevant queries, documents, and collaborators. We propose a graph algorithmic model to study such relationships. Based on this model, a novel *FlowRank* algorithm translates the collaborative ranking problem into a network flow problem. The heterogeneous interactions between queries, documents, and collaborators are drawn into a concise and cohesive framework. Our paper makes the following contributions:

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 434 – 445, 2006.
© Springer-Verlag Berlin Heidelberg 2006

- We introduce a formal model of collaborative ranking, which mathematically describes the interactions between users, queries, and documents, and relates the ranking problem with the network flow problem for which there is a large body of work.
- We propose a practical algorithm for collaborative ranking based upon our model, and empirically demonstrate its potential in a preliminary experiment using real-world data.

The rest of the paper is organized as follows[1]. In Section 2, we briefly review a number of relevant studies in collaborative ranking and graph modeling of the Web. We describe in detail the algorithmic model and propose a derived ranking algorithm, *FlowRank*, in Section 3. In Section 4, we present an evaluation involving realistic search scenarios. We conclude our paper with plans for future work in Section 5.

2 Related Work

There has been a considerable amount of work on the topic of collaborative search and ranking. We will discuss succinctly the major body of work that directly relate to ours, in particular to graph models of the Web and collaborative ranking algorithms.

The Web graph [1, 2] has been explored with numerous studies focusing on the hyper-linked structure in order to aid web search [3, 4, 5, 6, 7]. Flow-based algorithms have been proposed to identify and mining online communities [8, 9], to perform clustering [32], and to identify the bottlenecks in a Markov decision process [10]. In particular, Chitrapura and Kashyap [11] proposed a flow-based model for document ranking, which uses the network flows in a search graph as a measurement of relevance. In their model, the volume of the flows indicating the degree of relevance of the nodes (documents) to the associated labels (queries) is used to compute a query dependent or independent ranking of the documents, which is similar to the underlying idea we propose. However, their model is a single-user model, while our model employs multiple collaborating users with various degrees of similarity to the target user and different relevance feedbacks.

Collaborative search and ranking is in a way similar to meta-search; leveraging naïve and advanced users, the original search results are re-ranked based on collective knowledge. The search system can take into account the expertise level of the users through user profiling [12], biasing the relevance ranking by using such profiles. When a user is not satisfied with the search results, similar queries submitted by other users can be used to expand or improve the original query [13]. User clickthrough has been shown as an accurate reflection of users' preferences on the retrieved pages [14], to suggest search intentions [15] and similar queries and documents [16]. A hit matrix, which records users' clickthrough to pages, is implemented in the I-SPY system [17]. By using collective clickthrough as an indicator of the likeliness of web pages to be visited by users, the system estimates the pages' relevance to a given query, and re-ranks the pages based on the learned preferences. One of the most recent evaluations of collaborative search involves studying the behavior of actual users [18].

[1] Due to space constraints, an extended version of this paper discussing the generalization of the proposed model is available for download on the first author's website.

Our work builds upon the idea of graph modeling of the Web search problem, in a framework of relevance ranking of documents based on the associated flows of information. However, to the best of our knowledge, there is no existing work that is similar to our approach of constructing the graph model and the derived collaborative ranking algorithm.

3 The *FlowRank* Model

3.1 Motivation

Consider the following scenario of a typical collaborative search: A user u_a searches the Web for query q_o '*Olympic national park*', and a document set D_o has been retrieved. Then, suppose there is a *collaborator* u_b who searches for a relevant query q' '*camping hiking Olympic*', and a document set D' has been retrieved. We define collaborators to be those users whose queries and search results may be of interests to u_a with respect to u_a's current search goal, regardless of whether they perform the search in a synchronous or asynchronous fashion, or whether they are aware of each other. How relevant u_b's query is to u_a, and how similar u_b is to u_a in terms of interests and preferences, should be considered in the case of collaborative search. If u_b frequently accesses (which may mean, for example, that the user clicks on) some of the returned documents, it could indicate that these documents are of high quality and are preferable to other documents. Thus u_a might also prefer these on the condition that u_b's query is somewhat similar and u_b has similar interests or preferences.

From the above scenario, we see that a complete model for collaborative ranking needs to consider all of the following: the similarity between the original query (q_o) and the retrieved documents (D_o), the similarity between the original query (q_o) and other relevant queries (q'), the similarity between the relevant queries (q') and the corresponding documents (D'), the similarity between the user (u_a,) and the collaborators (u_b), and the access patterns of the users. All of these factors combined could contribute to the relevance judgment of the documents retrieved for a given query, and can be employed in a collaborative scheme of ranking those documents.

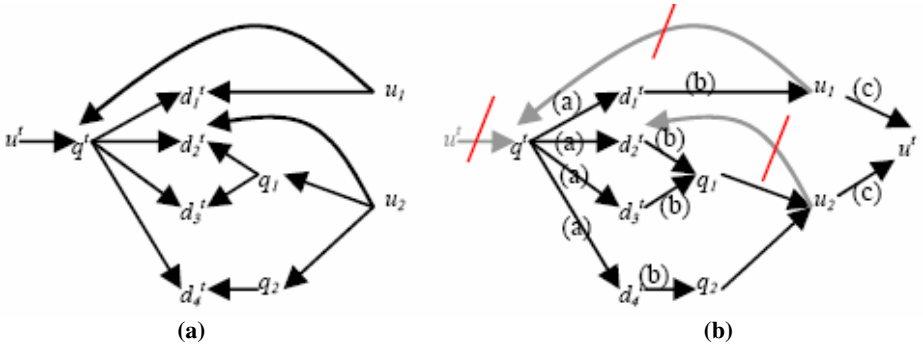
We have devised a formal model of collaborative ranking, in which all these factors are accounted for in a unified formal framework that permits the study of correlated search events. We accomplish this by investigating the network flows of a transformed query graph. In our model, each retrieved document is represented by a sink node in the graph and associated with a flow value. The flow values are bounded by the various capacities of the arcs, which correspond to the factors discussed in the previous paragraph (details to follow in the next section). When the flow network becomes saturated, the values of the obtained flows associated with the documents are used to re-compute the collaborative relevance ranking.

3.2 Graph Definition and Transformation

A query graph visualizes relevant entities (queries, documents, and users) in a given collaborative search setting. We transform this graph into a network in which arc capacities encode the relationships between the entities. In this section we formally describe the steps to derive the graph model. An algorithm for collaborative ranking based on this model will be introduced in the next section.

We first discuss the graph structure that we associate with a user query, which links users, queries, and documents sets, denoted as U, Q, and D respectively.

Assume that a target user u^t, submits a target query q^t, for which a set of documents $D^t=\{d_i^t \mid i=1..m\}$ matching the target query exists. Let $Q^t = \{q_i \mid \exists\, j \in [1\ldots n], arc(q_i, d_j^t) \neq \text{null}\}$be the set of all user queries that retrieve at least one of the documents retrieved by the target query q^t. Let $U^t = \{u_i \mid \exists\, q \in Q^t, arc(u_i, q) \neq \text{null}\}$denote the set of users who submitted the query that retrieved a document in D^t. Let G denote the whole search graph with no constraint. We denote by G_t the subgraph of G with the vertices $D^t \cup Q^t \cup U^t$ and all arcs incident in these vertices from G. Intuitively, the graph G_t includes those users who sent to the search engine the same query or a different query that retrieves at least one of the documents retrieved by the target query. Users who have not submitted q^t and queries that do not retrieve any document in D^t will not be included in this graph. The goal is to rank collaboratively the documents in D^t by considering the *appropriately relevant* users, queries, and documents. Note that in practice, there can be various definitions for $q_i \in Q^t$. For example, q_i can be an expansion of the target query q^t [19] or a query submitted by a user that searched for q^t in the same search session and/or within a certain interval of time. Figure 1 (a) depicts a sample query graph G_t.



(a) **(b)**

Fig. 1. (a) A sample query graph G_t (b) An example of G' constructed from G_t

We then construct G' based on G_t. Figure 1 (b) depicts G' constructed from G_t. There are three types of capacities assigned to the arcs in G': type (a) expresses the relevance between the target query and the documents; type (b) reflects various degrees of collective endorsement to the documents by the collaborators, either through the original query or alternative queries; type (c) specifies the relationship between the targeted user and his collaborators. **In particular:**

- Some arcs have their directions reversed in order to form the cascade of bounds for the network flows, i.e. type (a) capacities are more important than type (b); type (b) are more important than type (c). This is arguably reasonable because in ranking the relation between the target query and the documents should always be considered first, while the collective endorsement by the collaborators is counted as an assistive measure. Please see the following algorithm for details on which arcs to be reversed.

- Steps (2)~(6) of the following algorithm provide a specific computation recipe of the arc capacities. However, alternative strategies/heuristics can be employed to weigh the relationships among the different entities.

Let $U' = U^t - \{u^t\}$ denote all users in this graph except the target user and $Q' = Q^t - \{q^t\}$ the set of all user queries in G_t excluding the target query q^t. We construct G' as follows:

(1) Remove the arc from u^t to q^t. The relationship between u^t and q^t will be implicitly expressed in the generated flow values (defined in Section 3.3).
(2) Add an arc between any $u_i \in U'$ and u^t, and define its capacity $C(u_i, u^t)$ to be a similarity score between the two users, $0 < C(u_i, u^t) \leq 1$. This places an upper-bound for the flow values proportional to the user-user similarity.
(3) Define the capacity between q^t and any $d_k \in D^t$ to be a value in [0, 1] that indicates a matching score between the target query q^t and the document d_k. This places an upper-bound for the flow values proportional to the query-document similarity.
(4) For any $u_i \in U'$ and $q_j \in Q'$, if $arc(u_i, q_j) \neq null$ (meaning that the user u_i submitted query q_j), reverse the direction of this arc and set the capacity $C(q_j, u_i) = 1$, so that the flow value on $arc(q_j, u_i)$ is bounded *only* by the *upstream* capacities (i.e. query-document similarity), because of the reversed arc direction. Otherwise, set the capacity $C(q_j, u_i) = 0$. At this point we do not consider the frequency of u_i submitting query q_j.
(5) For any arc $arc(q_j, d_k)$ from q_j in Q' to a document $d_k \in D^t$, reverse the direction of this arc and define the capacity $C(d_k, q_j)$ to be $C(d_k, q_j) = sim(q_j, d_k) \cdot sim(q_j, q^t) \cdot P(d_k \mid U_{j,k})$, where $sim(q_j, d_k)$ in [0, 1] indicates a matching score between the query q_j and the document d_k; $sim(q_j, q^t)$ indicates how similar this alternative query q_j is to the target query q^t; $P(d_k \mid U_{j,k})$ indicates the conditional probability of users visiting d_k given that they submitted the query q_j, which can also be written as:

$$C(d_k, q_j) = sim(q_j, d_k) \cdot sim(q_j, q^t) \cdot \frac{Click(U_{j,k}, d_k)}{Click(U_{j,k}, D^t)}$$

$U_{j,k} = \{u_i \in U' \mid arc(u_i, q_j) \neq null \wedge arc(q_j, d_k) \neq null\}$. Note that $U_{j,k}$ typically contains more than one user; $Click(U_{j,k}, d_k)$ is the total number of clicks made by the users who submitted q_j on d_k; $Click(U_{j,k}, D^t)$ is the total number of clicks of these users on the whole search result set. $C(d_k, q_j)$ places an upper-bound for the flow value jointly decided by the similarity between an alternative query q_j and the target query q^t, the matching score between q_j and the document d_k, and the likelihood that a user visits d_k given that he submits q_j.
(6) For any $arc(u_i, q^t)$ for which $\exists d_k \in D^t$, $arc(u_i, d_k) \neq null$, reverse the direction of $arc(u_i, d_k)$ and assign $C(d_k, u_i) = sim(q^t, d_k) \cdot P(d_k \mid u_i) \cdot [1 - P(d_k \mid \overline{\{u_i\}})]$, where the last two factors represent the probability of a user u_i visiting d_k and the conditional probability of the *other* users *not* visiting d_k given that they submitted the target query q^t. This can also be written as:

$$C(d_k, u_i) = sim(q^t, d_k) \cdot \frac{Click(u_i, d_k)}{Click(u_i, D^t)} \cdot \left(1 - \frac{Click(\overline{\{u_i\}}, d_k)}{Click(\overline{\{u_i\}}, D^t)}\right)$$

where $\overline{\{u_i\}} = U' - \{u_i\}$; $Click(u_i, d_k) = 1$ if u_i clicked on d_k and 0 otherwise; $Click(u_i, D^t)$ is the number of documents clicked by u_i. $C(d_k, u_i)$ places an upper-bound for the flow value jointly decided by the matching score between the target query q^t and the document d_k, and the likelihood that other users also visit d_k given that they also submit q^t.

(7) Remove all the arcs from $u_i \in U'$ to q^t. Because the inherent relationship is already reflected in the definition of $C(d_k, u_i)$, these arcs are redundant and do not contribute to the collaborative ranking process.

(8) END

Finally, we want to provide an intuitive justification for constructing G'. A query represents the information needs of a user, and the motivation of search is to satisfy such needs. In G', flows of relevant information are moving through a network of collaboration towards the target user via different routes. Each of these flows contributes as a part to the overall information gain pertinent to the user's needs. On each of the routes, there are an appropriately relevant document, an appropriately relevant query, and an appropriately relevant user; all of them together determine the contribution of the associated flow. Flows, when saturated, represent the collaborative contribution to the overall satisfaction of the target user's information needs. By investigating G', we are able to study the relationship between the collaborative entities when the flows are sorted in the order of their contributions to the overall satisfaction of the target user's information needs. Accordingly the associated documents en route can also be ranked using the *FlowRank* algorithm described in the next section.

3.3 The *FlowRank* Algorithm

Let s denote the source and t denote the sink, $F_{max}(s, t)$ is defined as the maximum flow that can be routed from s to t, which obeys all the capacity constraints. Intuitively, if the arcs are water pipes, the vertices are where they join each other, and the capacities on the arcs represent the cross-sectional area of the pipes, to find the maximum flow is to find how much water can be moved from s to t, given the constraints of the cross-sectional area of the pipes. The *FlowRank* algorithm is described in Figure 2.

The *FlowRank* algorithm is based on the well-known Maximum Flow - Minimal Cut theorem [20]. This theorem proves that the maximum flow of a given network is equal to the minimal cut that separates the source and the sink, which in our case the cut is D^t, proved as follows. A set of cut-vertex denotes the set of vertexes whose removal will disconnect the graph [30]. By assumption that D^t is not the set of cut-vertex on G', we remove D^t and its related arcs, and still have q^t connected to G' via a set of vertex V_{qt}. $\forall v_{qt} \in V_{qt}$: $v_{qt} \notin Q$, because by definition there's no arc exists between two queries; $v_{qt} \notin U$, because of step (4) and (5) in 3.2; thus $v_{qt} \notin D$. Because $D = D^t \cup D'$ and D' is removed in step (1) in 3.2, $v_{qt} \in D^t$. But this contradicts the assumption that D^t has already been removed from G'. Q.E.D.

The above proof is important for the validity of the algorithm in that $\forall F(q^t, u^t)$, $\exists$ $d_i^t \in D^t$ which is on the path of $F(q^t, u^t)$. Because D^t is not inter-connected, we conclude the there is d_i^t on each of the paths of $F(q^t, u^t)$, so that D^t can be ranked by sorting $F(q^t, d_i^t)$.

Algorithm. FlowRank

Input: Graph G'

Output: Permutation π { $\pi(d_1{}^t),...,\pi(d_m{}^t)$ }, $d_1{}^t \ldots d_m{}^t \in D^t$.

1: $F_{max}(q^t, u^t) \leftarrow$ the maximum flow from source q^t to sink u^t

2: **foreach** document $d_i{}^t \in D^t$

3: $F(q^t, d_i{}^t) \leftarrow$ the flow value from source q^t to sink $d_i{}^t$

4: Sort (descending) the documents $d_i{}^t$ in D^t using $F(q^t, d_i{}^t)$

5: **return** ranks $\left\{\pi(d_1{}^t),...,\pi(d_m{}^t)\right\}$

Fig. 2. The *FlowRank* Algorithm

4 Evaluation

4.1 Experiment Setup

From the query logs of a popular scientific document search engine, The *CiteSeer* Digital Library (http://citeseer.ist.psu.edu), we extracted a random sample of 100 queries and the associated 1,334 unique documents retrieved by the search engine as relevant results. Obtaining editorial ratings for a large number of documents is an extremely time-consuming and labor-intensive process, however the size of our dataset is comparable to those used in [21]. The queries were anonymized, and then verified by human annotators to be *meaningful* [22].

Our evaluation required a log of user interactions (i.e. clickthroughs, etc.) with the documents. The extracted queries and documents were presented to five evaluators, all graduate students in Computer Science. For each document, the title and abstract/snippet were displayed. In order to minimize the potential bias induced by ranking, for each query the associated documents were shuffled so that the evaluators were not aware of the original ranking produced by the search engine, and the evaluators were explicitly informed about this process. The evaluators were asked to independently choose one of the three ratings for each document based on their subjective perception of how relevant a document is to the associated query. The three ratings were: "*Definitely clicked*", which means the evaluator believes he/she would click on the document if he/she were to submit the query; "*Probably clicked*", which means the evaluator may or may not click on the document, and "*Never clicked*", which means the evaluator believes he/she would not click on the document. The data collected in this procedure is labeled as D_s. Ideally we would like to keep track of a user's clicks on a document, denoted by a 4-tuple (*q_id, doc_id, u_id, p_click*), where *q_id* and *doc_id* denote the query and corresponding documents, *u_id* denotes the user, and *p_click* denotes the probability of the user clicking on the document. In

processing D_s, the rating "Definitely clicked" was translated into *p_click*=1.0; "Probably clicked" was translated into *p_click*=R where R is a random variable within (0, 1); "Never clicked" was translated into *p_click*=0.0. The obtained data D_s was then used to simulate user clickthrough.

Relevance judgments for the dataset were generated by presenting the same queries and documents to another two evaluators who are both computer scientists. This time, the evaluators could check the actual content of the document when necessary. Each of the evaluators was asked to rate the document in a five-point scale from 0 to 4, defined as:

0. *Irrelevant match*: the document did not contain any information about the query.
1. *Marginally relevant match*: the query terms might appear in the document but it was mainly about something else.
2. *Borderline match*: the document could be rated as 1 or 3.
3. *Fairly relevant match*: the document contained relevant information about the query terms, but should not be picked if only one document were allowed.
4. *Best match*: the document contained highly relevant information about the query terms, and could be picked if only one document were allowed.

Data collected in this phase is labeled as D_e.

A number of measures have been proposed to quantitatively describe the similarity between queries [15, 24, 25, 27]. In our implementation, the definition of similarity between queries was similar to the *common query title* measurement as described in [25]:

$$sim(q_i,q_j)=\frac{\left|T_i \cap T_j\right|}{\max(\left|T_i\right|,\left|T_j\right|)}$$

where q_i, q_j were two queries and T_i, T_j were the terms in the titles of the documents returned by the search engine. In other words, the similarity between the two queries was in proportion to the number of common terms in the titles of the search results. If the titles were similar, the queries were also similar.

User similarity metrics are commonly adopted in collaborative recommendation and filtering systems, because by learning from other "like-minded" users, one can predict a user's preference (see [25] for a recent survey). User similarity is inherent in our model because of step (2) of the algorithm. The user similarity score is the upper-bound for the flow values. For simplicity of computation, we currently treat all users the same, so for any two users u_i and u_j, $sim(u_i, u_j) = 1$. We justify this assumption by noting that all of the five evaluators were graduate students in Computer Science and most likely familiar with the search topics. However, it would be interesting to observe the impact of different user similarity metrics in the next phase of evaluation.

Finally, the similarity scores between the queries and documents were derived from the search engine. Now a transformed query graph was computed for each of the queries, and the *FlowRank* algorithm was implemented on the graphs to generate the collaborative rankings. Adjacent matrixes were built to represent the transformed query graphs, with elements being the arcs' capacities. In our experiment, the maximum-flow calculation module was based on Rothberg's implementation of Goldberg's Push-Relabel algorithm [23], which is usually considered the fastest in practice.

4.2 Results and Discussion

The output of *FlowRank* (*F*) was compared with that of other two baseline ranking algorithms for evaluation. The HITS algorithm [4] was implemented and the ranking (*H*) for the documents was recorded. The original ranking (*S*) generated by *CiteSeer* was also recorded. The metrics to measure the ranking accuracy was the Discounted Cumulative Gain (DCG) first introduced in [26] and compared to other metrics in [28]. The main reason to use DCG is that it assigns more weights to highly ranked documents, and allows us to differentiate various levels of subjective relevance judgment for the human evaluators. For a given query *q*, DCG is defined as:

$$DCG(q,N) = \sum_{d=1}^{N} \frac{\left(2^{R(d)} - 1\right)}{\ln(1+d)} \tag{1}$$

where *R(d)* is the editorial rating of the d'th webpage in the top *N* search results.

Intuitively, a higher DCG reflect a better ranking of the search results. DCG was computed for the top 20 ranked documents in *H*, *S*, and *F*, since users seldom look beyond the first few result pages [31]. The ratings from the two evaluators in D_e were averaged as the editorial ratings to be used in the DCG calculation.

FlowRank achieved significant improvements on the rankings produced by the two baseline algorithms. This advantage was reflected in the average DCG metrics. In both comparisons, *FlowRank* was able to improve the ranking for about 47% of the queries. Over all ranks, the average DCG for *F* was 59.42, compared with 57.54 for *S* and 57.37 for *H*. The number of queries with DCG increased, decreased, and without change was summarized in Figure 3(a).

Figure 3(b) plots the average DCG curves for the three ranking algorithms (*F, H, S*) at ranks 1 – 20, together with the ideal curve. *FlowRank* outperformed the other two baseline algorithms and quickly approached the ideal curve, which began to level off upon the rank 10. This confirmed that the documents ranked below position 10 were in fact less relevant.

In order to further investigate if there is linear correlation between the collaborative contribution considered by *FlowRank*, and the amount of increase in the average DCG over the two baseline algorithms, the improvement in the DCG scores for each of the queries was cross-examined with the size of the transformed graph. Here an assumption was made that the larger the transformed query graph, the more collaborative the search process would be. A correlation test was performed on the two variables: size *S* and increase in the DCG *I*. The Pearson correlation of *S* and *I* was 0.334 with a P-value of 0.033 (α-level was 0.05), confirming a positive correlation between the two.

A simplification was drawn in the current implementation of the flow-based model, in which the arc capacities between $\{u_i'\}$ and u_t were assigned the same values, indicating identical collaborators. Three runs of the *FlowRank* algorithm were performed in which the number of users was purposefully changed. Correlated changes in the DCG improvement had been observed. The correlation was not significant, which we believed would be mostly due to the small number of users.

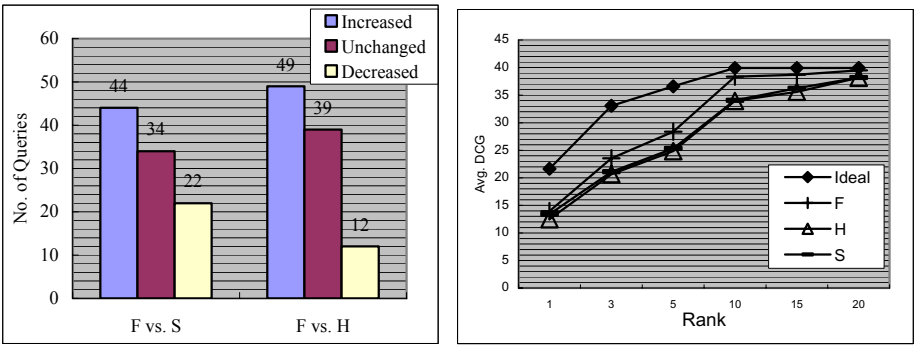


Fig. 3. (a) DCG changes for queries; *FlowRank* was able to improve the search ranking for about 47% of the queries. **(b)** Average DCG curves; The *H* and *S* curves hang below the *F* curve by about 15%.

The maximum flows were currently calculated using the Push-Relabel algorithm [23], because the sizes of the graphs were relatively small, and runtime efficiency was a primary concern due to the online nature of collaborative ranking. However, depending on the scale (and often the subject domain) of a collaborative search, the graph can become very dense or sparse, or can become so huge that access to the entire graph is impractical. Different maximum-flow algorithms, e.g. the shortest augmentation path algorithm [29], should be used with case-by-case consideration.

5 Conclusion and Future Work

We proposed a comprehensive flow-based graph model of collaborative ranking. Exploiting the relationships between relevant queries, documents, and users, and users' access patterns on the retrieved documents, this model translates the collaborative ranking problem into a flow-calculation problem in a search network. This unique perspective considers the ranking problem as a graph flow problem for which there is a large body of work. We discussed the implications of several ranking scenarios and presented a derived practical ranking algorithm. Evaluations in the document search domain using the DCG metrics showed its effectiveness by measurable improvements over the two baseline ranking algorithms.

There are a number of directions towards which this work can be applied. Future work can explore different similarity metrics and use the *FlowRank* model to quantitatively investigate the complicated relationships between queries, documents, and users in a collaborative search setting. Obtaining editorial ratings for a large number of documents is an extremely time-consuming and labor-intensive process, which limited the scale of our preliminary experiment. A reasonable next phase of evaluation is to compare *FlowRank* against other collaborative ranking algorithms using a large-scale dataset. Instead of using simulated clickthrough data, real-world clickthrough logs can be used so that the definitions of the arc capacities can be further fine-tuned. Additionally, it may be valuable to also take into account user

interactions other than clickthroughs, such as how long a user spent reading a document. Given the generalizable nature of the proposed network flow model, we believe that it can be applied to a number of other problems dealing with user feedback and collaborative search behavior.

Acknowledgments. We gratefully acknowledge E. Brill, C. Burges, M. Richardson, and the anonymous reviewers for their comments, Y. Sun for his help on the experiments, and partial support from the National Science Foundation.

References

1. Bharat, K., Broder, A., Henzinger, M., Kumar, P., and Venkatasubramanian, S. The Connectivity Server: Fast Access to Linkage Information on the Web. In Proc. of the 7th International World Wide Web Conference, pp. 469-477, 1998.
2. Broder, A., Kumar, R., Maghoul1, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. Graph structure in the web. In Proc. of the 9th International World Wide Web Conference, pp. 309-320, 2000.
3. Carriere, J., and Kazman, R. WebQuery: Searching and visualizing the Web through connectivity. In Proc. of the 6th International World Wide Web Conference, 1997.
4. Kleinberg, J. Authoritative sources in a hyperlinked environment. J. of the ACM, Vol. 46, Issue 5, pp. 604-632, 1999.
5. Brin, S., and Page, L. The anatomy of a large scale hypertextual web search engine. In Proc. of the 7th International World Wide Web Conference, pp. 107-117, 1998.
6. Chakrabati, S., Dom, B., Gibson, D., Kleinberg, J, Raghavan, P., and Rajagonpalan, S. Automatic Resource List Compilation by Analyzing Hyperlink Structure and Associated Text. In Proc. of the 7th International World Wide Web Conference, 1998.
7. Tomita, J., and Kikui, G. Interactive Web Search by Graphical Query Refinement. In Proc. the 10th International World Wide Web Conference, 2002.
8. Flake, G. W., Lawrence, S., and Giles, C. L. Efficient identification of web communities. In Proc. of the 6th International Conference on Knowledge Discovery and Data Mining, pp. 150-160, 2000.
9. Flake, G. W., Tsioutsiouliklis, K., and Zhukov, L. Methods for Mining Web Communities: Bibliometric, Spectral, and Flow. Web Dynamics, Springer Verlag, 2003.
10. Menache, I., Mannor, S., and Shimkin, N. Q-Cut - Dynamic Discovery of Sub-goals in Reinforcement Learning, In Proc. of the 13th European Conference on Machine Learning (ECML), pp. 295-306, 2002
11. Chitrapura, K. P., and Kashyap S. R. Node Ranking in Labeled Directed Graphs. In Proc. of ACM Conference on Information and Knowledge Management, pp. 597 - 606. 2004.
12. Chidlovskii, B., Glance, N. S., and Grasso, M. A. Collaborative Re-Ranking of Search Results. In The National Conference on Artificial Intelligence 2000 Workshop on AI for Web Search, pp. 18 - 23. 2000.
13. Zaiane, O. R., and Strilets, A. Finding Similar Queries to Satisfy Searches based on Query Traces. In Proc. of the International Workshop on Efficient Web-Based Information Systems, pp 207-216, 2002.
14. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G. Accurately Interpreting Clickthrough Data as Implicit Feedback. In Proc. of Annual ACM Conference on Research and Development in Information Retrieval (SIGIR) 2005, pp. 154-161, 2005.

15. Daume, H., and Brill, E. Web search intent induction via automatic query reformulation. In Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics, 2004.
16. Wen, J., Nie, J. and Zhang. H. Clustering user queries of a search engine. In Proc. of the 10th International World Wide Web Conference, pp. 162-168. 2001.
17. Freyne, J., Smyth, B., Coyle, M., Balfe, E., Briggs, P. Further Experiments on Collaborative Ranking in Community-Based Web Search. Artificial Intelligence Review, Vol. 21:3-4, pp. 229-252. 2004.
18. Smyth, B., Balfe, E., Boydell, O., Bradley, K., Briggs, P., Coyle, M., Freyne, J. A Live-User Evaluation of Collaborative Web Search. In Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05). 2005.
19. Vakkari, P. Subject knowledge, source of terms and term selection in query expansion. In Proc. of the 24th European Conference in Information Retrieval, 2002.
20. Ford, L. R. Jr., and Fulkerson, D. R. Maximal flow through a network. Canadian J. of Mathematics, Vol. 8, pp. 399-404, 1956.
21. Savoy, J. and Vrajitoru, D. Evaluation of learning schemes used in information retrieval. Technical Report CR-I-95-02, Faculty of Sciences, University of Neuchatel, 1996.
22. Risvik, K. M., Mikolajewski, T., and Boros, P. Query Segmentation for Web Search. In Proc. of the 11th International World Wide Web Conference. May 20-24, 2003.
23. Goldberg, A. V., and Tarjan, R. E. A new approach to the maximum-flow problem. J. of the ACM, Vol. 35, Issue 4, pp. 921-940. ACM Press, 1998.
24. Beeferman, D., and Berger, A. Agglomerative clustering of a search engine query log. In Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 407-416, 2000.
25. Cheung, K., and Tian, L. Learning User Similarity and Rating Style for Collaborative Recommendation. Information Retrieval, Vol. 7: 3-4, pp. 395-410. Srpinger-Verlag, 2004.
26. Jarvelin, K., and Kekalainen, J. IR Evaluation Methods for Retrieving Highly Relevant Documents. In Proc. of the 23rd Annual ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 41-48. 2000.
27. Chien, S., Immorlica, N. Semantic Similarity Between Search Engine Queries Using Temporal Correlation. In Proc. of the 14th International Conference on World Wide Web, pp. 2-11. 2005.
28. Jarvelin, K., and Kekalainen, J. Cummulated Gain-based Evaluation of IR Techniques. In ACM Transactions on Information Systems (TOIS), Vol. 20, No. 4, pp. 422-446, 2002.
29. Edmonds, J., Kapr, R. M. Theoretical improvements in the algorithmic efficiency for network flow problems. J. of ACM, Vol. 19, pp. 248-264. 1972.
30. Chartrand, G. Cut-Vertices and Bridges. Introductory Graph Theory. New York: Dover, pp. 45-49, 1985.
31. Jansen, J., and Spink, A. An Analysis of Web Documents Retrieved and Viewed, In Proc. of the 4th International Conference on Internet Computing, 2003.
32. Flake, G. W., Tarjan, R. E., and Tsioutsiouliklis, K. Graph Clustering and Minimum Cut Trees. J. of Internet Mathematics, Vol. 1, No. 4, pp. 385-408. 2004.

Finding Hierarchies of Subspace Clusters

Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, Ina Müller-Gorman, and Arthur Zimek

Institute for Informatics, Ludwig-Maximilians-Universität München, Germany
{achtert, boehm, kriegel, kroegerp, muellerg, zimek}@dbs.ifi.lmu.de

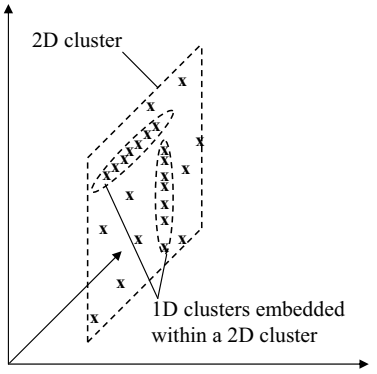
Abstract. Many clustering algorithms are not applicable to high-dimensional feature spaces, because the clusters often exist only in specific subspaces of the original feature space. Those clusters are also called subspace clusters. In this paper, we propose the algorithm HiSC (Hierarchical Subspace Clustering) that can detect hierarchies of nested subspace clusters, i.e. the relationships of lower-dimensional subspace clusters that are embedded within higher-dimensional subspace clusters. Several comparative experiments using synthetic and real data sets show the performance and the effectivity of HiSC.

1 Introduction

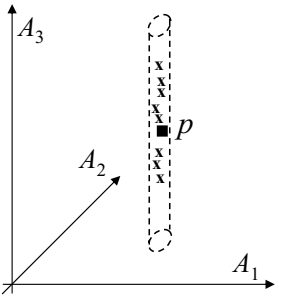
In high-dimensional feature spaces, many clustering algorithms are not applicable because the clusters often exist only in specific subspaces of the original feature space. This phenomenon is also often called *curse of dimensionality*. To detect such lower-dimensional subspace clusters, the task of subspace clustering (or projected clustering) has been defined recently. We will refer to a subspace cluster associated to a λ-dimensional projection/subspace (i.e. spanned by λ attributes) as a *λ-dimensional subspace cluster*. The dimensionality of a subspace associated to a subspace cluster is called *subspace dimensionality*.

In this paper, we focus on a second class of algorithms that assign each object to a unique cluster (or noise) rather than algorithms that allow overlapping subspace clusters. Existing algorithms for non-overlapping subspace clustering usually have one severe limitation in common. In case of hierarchically nested subspace clusters, i.e. several subspace clusters of low dimensionality may together form a larger subspace cluster of higher dimensionality, these algorithms will miss important information about the clustering structure. For example, consider two axis-parallel lines in a 3D space that are embedded into an axis-parallel 2D plane (cf. Figure 1(a)). Each of the two lines forms a 1-dimensional subspace cluster. On the other hand the plane is a 2-dimensional subspace cluster that includes the two 1-dimensional subspace clusters. In order to detect the lines, one has to search for 1-dimensional subspace clusters, whereas in order to detect the plane, one has to search for 2-dimensional subspace clusters. Moreover, searching subspace clusters of different dimensionality is essentially a hierarchical problem because the information that a point belongs e.g. to some k-dimensional subspace cluster that is itself embedded into an l-dimensional subspace cluster ($k < l$) can only be uncovered using a hierarchical approach.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 446–453, 2006.
© Springer-Verlag Berlin Heidelberg 2006



(a) Hierarchically nested subspace clusters.



(b) Subspace dimensionality of a point (see text for details).

Several subspace clustering algorithms aim at finding all clusters in all subspaces of the feature space. Those algorithms produce overlapping clusters where one point may belong to different clusters in different subspaces. Well-known examples of such algorithms include e.g. CLIQUE [1], ENCLUS [2], SUBCLU [3], and FIRES [4].

Here, we focus on finding non-overlapping subspace clusters, i.e. assigning each point to a unique subspace cluster or noise. The probably most prominent examples for subspace clustering algorithms producing non-overlapping clusters are e.g. PROCLUS [5], DOC [6], and PreDeCon [7].

However, none of the proposed approaches to subspace clustering can detect nested hierarchies of subspace clusters. Thus, in this paper, we propose HiSC (Hierarchical Subspace Clustering), a new algorithm that applies a hierarchical approach to subspace clustering and, thus, detects hierarchies of subspace clusters.

2 Hierarchical Subspace Clustering

Let $\mathcal{D}$ be a data set of n feature vectors of dimensionality d ($\mathcal{D} \subseteq \mathbb{R}^d$). Let $\mathcal{A} = \{a_1, \ldots, a_d\}$ be the set of all attributes a_i of $\mathcal{D}$. Any subset $S \subseteq \mathcal{A}$, is called a *subspace*. The *projection* of an object $o \in \mathcal{D}$ into a subspace $S \subseteq \mathcal{A}$ is denoted by $\pi_S(o)$. The distance function is denoted by $dist$. We assume that $dist$ is one of the L_p-norms.

The aim of HiSC is to detect clusters of lower dimensional subspaces contained in clusters of higher dimensional subspaces. Our general idea is to evaluate whether two points are contained in a common subspace cluster. For example, two points that are in a 1D subspace cluster may be contained in a 2D cluster that consists of the two 1D projections. We perfom this evaluation with a special distance measure called *subspace distance*. This distance results in a small value whenever two points are in a common low-dimensional subspace cluster, whereas the subspace distance is high if both points are in a common high-dimensional subspace cluster or are not in a subspace cluster at all. Therefore, our strategy is to merge those points into common clusters which have

small subspace distances. A hierarchy of subspace clusters means that clusters with small subspace distances are nested in clusters with higher subspace distances.

In order to define the already mentioned subspace distance, we assign a subspace dimensionality to each point of the database, representing the *subspace preference* of its local neighborhood. The subspace dimensionality of a point reflects those attributes having a small variance in the local neighborhood of the point. As local neighborhood of a point p we use the k-nearest neighbors of a point p, denoted by $\mathrm{NN}_k(p)$. The *variance* of the local neighborhood of a point $p \in \mathcal{D}$ from p along an attribute $A_i \in \mathcal{A}$, denoted by $\mathrm{VAR}_{A_i}(\mathrm{NN}_k(\mathrm{p}))$, is defined as follows:

$$\mathrm{VAR}_{A_i}(\mathrm{NN}_k(p)) = \frac{\sum_{q \in \mathrm{NN}_k(p)} (\pi_{A_i}(q) - \pi_{A_i}(p))^2}{|\mathrm{NN}_k(p)|}.$$

Intuitively, the subspace dimensionality is the number of attributes with high variance. Similar to [7], we assign a subspace preference vector to each point, indicating attributes with high and low variance.

Definition 1 (subspace preference vector of a point)
Let $\alpha \in \mathbb{R}$ be a threshold value. The subspace preference vector *of a point $p \in \mathcal{D}$, $w_p = (w_p^1, \ldots, w_p^d)^{\mathbf{T}}$, is defined as*

$$w_p^i = \begin{cases} 0 & \textit{if} \quad \mathrm{VAR}_{A_i}(NN_k(p)) > \alpha \\ 1 & \textit{if} \quad \mathrm{VAR}_{A_i}(NN_k(p)) \leq \alpha \end{cases}$$

The subspace dimensionality of a point can now be defined as follows.

Definition 2 (subspace dimensionality of a point)
The subspace dimensionality *λ_p of a point $p \in \mathcal{D}$ is the number of zero-values in the subspace preference vector of p, w_p, formally:*

$$\lambda_p = \sum_{i=1}^{d} \begin{cases} 1 & \textit{if} \quad w_p^i = 0 \\ 0 & \textit{if} \quad w_p^i = 1 \end{cases}$$

An example is visualized in Figure 1(b). The 9-nearest neighbors of the 3D point p exhibit a 1D subspace cluster spanned by the attribute A_3, i.e. the variance of the neighborhood of p is high along attribute A_3, whereas it is low along attributes A_1 and A_2. Consequently, $w_p = (1, 1, 0)^{\mathbf{T}}$ and $\lambda_p = 1$.

Once we have associated the points of our database to a (local) subspace dimensionality and to a subspace preference vector, we can now explain the main idea of our hierarchical clustering algorithm. Conventional hierarchical clustering algorithms like SLINK [8] or OPTICS [9] work as follows: They keep two separate sets of points, points which were already placed in the cluster structure and those which were not. In each step, one point of the latter set is selected and placed in the first set. The algorithm always selects that point which minimizes the distance to any of the points in the first set. By this selection strategy, the algorithm tries to extend the current cluster hierarchy as close to the bottom of the hierarchy as possible.

We will adapt this paradigm to the context of hierarchical subspace clustering where the hierarchy is a containment hierarchy of the subspaces. Two or more 1D subspace clusters may together form a 2D subspace cluster and so on. We simulate this behavior by defining a similarity measure between points which assigns a distance of 1, if these two points share a common 1D subspace cluster. If they share a common 2D subspace cluster, they have a distance of 2, etc. Sharing a common subspace cluster may mean different things: Both points may be associated to the same 2D subspace cluster, or both points may be associated to different 1D subspace clusters that intersect at some point or are parallel (but not skew).

If we assign a distance measure to a pair of points with the properties mentioned before, we can generally use the well-known hierarchical clustering algorithms. Intuitively, the distance measure between two points corresponds to the dimensionality of the data space which is spanned by the attributes of high variance of the neighborhoods of the two points. We first give a definition of the *subspace dimensionality of a pair of points* $\lambda(p, q)$ which follows the intuition of the spanned subspace. Then, we will define our subspace distance measure based on these concepts. In fact, the subspace dimensionality is the most important component of our distance measure.

Definition 3 (subspace preference vector/dimensionality of a pair of points)
The subspace preference vector $w(p, q)$ *of a pair of points* $p, q \in \mathcal{D}$ *representing the attributes with low and high variance of the combined subspace is defined by*

$$w(p, q) = w_p \wedge w_q \quad \textit{(attribute-wise logical AND-conjunction).}$$

The subspace dimensionality *between two points* $p, q \in \mathcal{D}$*, denoted by* $\lambda(p, q)$*, is the number of zero-values in* $w(p, q)$.

A first approach is defining the subspace distance between two points p and q as the subspace dimensionality $\lambda(p, q)$. We only need a slight extension for points that have the same subspace preference vector, but do not belong to the same subspace cluster. For these points, we have to check whether the preference vectors of two points are equal. If so, we have to determine the distance between the points along the attributes of low variance. If this distance, which can be evaluated by a simple weighted Euclidean distance using one of the preference vectors as weighting vector, exceeds α, the points (or corresponding clusters) do not belong to the same cluster but belong to different (parallel) clusters. The threshold α, playing already a key role in Definition 1, controls the degree of jitter of the subspace clusters.

As $\lambda(p, q) \in \mathbb{N}$, many distances between different point pairs are identical. Therefore, there are many tie situations during clustering. We resolve these tie situations by additionally considering the Euclidean distance within a subspace cluster as a secondary criterion. This means, inside a subspace cluster (if there are no nested lower-dimensional subspace clusters), the points are clustered in the same way as using a conventional hierarchical clustering method. The Euclidean distance between p and q hereby is weighted by the inverse of the combined preference vector $w(p, q)$ (as given in Definition 3). This inverse, $\bar{w}(p, q)$, weights the distance along attributes spanning the cluster with 1, the distance along any other attribute is weighted with 0. Formally we define:

Definition 4 (subspace distance)
Let $w = (w^1, \ldots, w^d)^{\mathbf{T}}$ *be a* d*-dimensional vector, and* $dist_w(p,q) = \sum w^i(\pi_{a_i}(p) - \pi_{a_i}(q))^2$ *be the weighted Euclidean distance w.r.t.* w *between two points* $p, q \in \mathcal{D}$*. The* subspace distance *between* p *and* q*, denoted by* $\text{SDIST}(p,q) = (d_1, d_2)$*, is a pair consisting of the following two values:*

$$d_1 = \lambda(p,q) + \begin{cases} 1 & \textit{if } \max\{dist_{w_p}(p,q), dist_{w_q}(q,p)\} > \alpha \\ 0 & \textit{else,} \end{cases}$$
$$d_2 = dist_{\bar{w}(p,q)}(p,q).$$

We say that $\text{SDIST}(p,q) \leq \text{SDIST}(r,s)$ *if* $\text{SDIST}(p,q).d_1 < \text{SDIST}(r,s).d_1$*, or* $\text{SDIST}(p,q).d_1 = \text{SDIST}(r,s).d_1$ *and* $\text{SDIST}(p,q).d_2 \leq \text{SDIST}(r,s).d_2$.

As discussed above, d_1 corresponds to the subspace dimensionality of p and q, taking special care in case of parallel clusters. The value d_2 corresponds to the weighted Euclidean distance between p and q, where we use the inverse of the combined preference vector, $\bar{w}(p,q)$, as weighting vector.

Using the subspace distance as defined in Definition 4 as a distance measure, we can basically run every hierarchical (or even non-hierarchical) clustering algorithm which is based on distance comparisons. Examples for such algorithms are Single-Link, Complete-Link, and the density-based method OPTICS [9]. HiSC computes a "walk" through the data set similar to OPTICS and assigns to each point o its smallest subspace distance with respect to a point visited before o in the walk. In each step of the algorithm, HiSC selects that point o having the minimum subspace distance to any already processed point. This process is managed by a seed list which stores all points that have been reached from already processed points sorted according to the minimum subspace distances. A special order of the database according to its subspace-based clustering structure is generated, the so-called *cluster order*, which can be displayed in a subspace distance diagram. Such a subspace distance diagram consists of the subspace distance values on the y-axis of all points, plotted in the order which HiSC produces on the x-axis. The result is a visualization of the clustering structure of the data set which is very easy to comprehend and interpret. The "valleys" in the plot represent the clusters, since points within a cluster typically have lower subspace distances than points outside of a cluster. The complete integration of our distance measure into the algorithm HiSC can be seen in Figure 1.

Input parameters. HiSC has two input parameters. First, the parameter k specifies the locality of the neighborhood from which the local subspace dimensionality of each point is determined. Obviously, this parameter is rather critical because if it is chosen too large, the local subspace preference may be blurred by noise points, whereas if it is chosen too small, there may not be a clear subspace preference observable, although existing. However, in our experiments, choosing k in the range between 10 and 20 turned out to produce very stable and robust results. Second, the parameter parameter α is important for specifying the attributes with low and high variance and, thus, α also affects the computation of the (local) subspace dimensionality of each point. In fact, attributes where the variance of the k-nearest neighbors of a point is below α are relevant for the subspace preference of the point. In our experiments, it turned out that

```
algorithm HiSC(𝒟, k, α)
    initialize empty priority queue pq   // ordered by SDIST
    for each p ∈ 𝒟 do
        compute w_p;
        p.SDIST = ∞;
        insert p into pq;
    while pq ≠ ∅ do
        o := pq.next();
        for each p ∈ pq do
            pq.update(p, SDIST(o, p));
        append o to the cluster order;
    return the cluster order;
```

Fig. 1. The HiSC algorithm

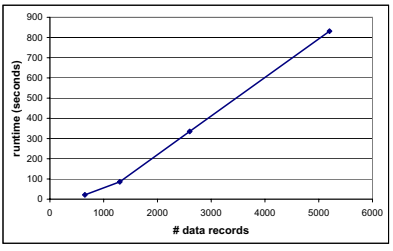
HiSC is quite robust against the choice of α, as long as α is chosen between 0.1% and 0.5% of the attribute range, i.e. the maximum attribute value. However, if one expects subspace clusters having a lot of jitter, α can be increased accordingly.

Runtime complexity. In the first loop the (local) subspace dimensionalities and preference vectors are precomputed which requires the determination of the k-nearest neighbors of each object. This step be done in $O(n \log n \cdot d)$ time assuming the use of a spatial index or in $O(n^2 \cdot d)$ withou index support. During the run of HiSC, we have to evaluate for each pair of points of the database its subspace dimensionality which is a simple logical AND-conjunction on the subspace dimensionality vectors and, thus, has a complexity of $O(d)$. Thus, the overall runtime complexity of HiSC is $O(n^2 \cdot d)$.

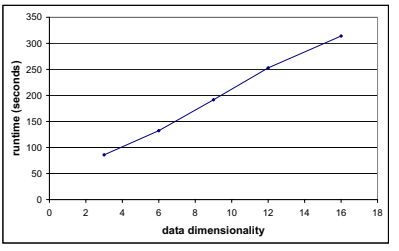
3 Experimental Evaluation

We evaluated the scalability of HiSC on a workstation featuring a 3.2 GHz CPU with 1GByte RAM. All parameters were choosen as suggested above. The scalability of HiSC w.r.t. the data set size is depicted in Figure 2(a). The experiment was run on a set of 3D synthetic data sets with varying number of records. Each data set contained two 1D clusters, two 2D clusters, and noise points. As it can be seen, HiSC scales nearly linearily w.r.t. the number of tuples in the dataset. A similar observation can be made when evaluating the scalability of HiSC w.r.t. the dimensionality of the data set (cf. Figure 2(b)). The experiments were obtained using data sets with 1,300 data points with varying dimensionality. Each data set contained two $(d-2)$-dimensional clusters, two $(d-1)$-dimensional clusters, and noise. Again, the result shows a linear increase of runtime when increasing the dimensionality of the data set.

We first evaluated HiSC on several synthetic data sets. Exemplary, we show the results on two data sets. Data set "DS1" (cf. Figure 3(a)) contains 3D points grouped in three hierarchical subspace clusters and noise. Two 1D clusters (cluster 1.1 and cluster 1.2) are both embedded within a 2D cluster (cluster 1). Data set "DS2" is a 20D data set containing three clusters of significantly different dimensionality and noise: cluster 1 is a 15D subspace cluster, cluster 2 is 10 dimensional, and cluster 3 is a 5D subspace cluster.

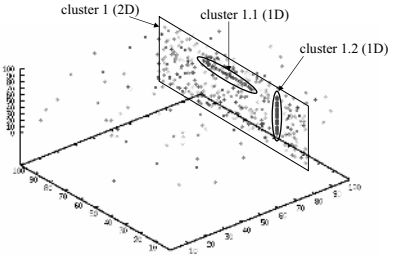


(a) Scalability w.r.t. size.

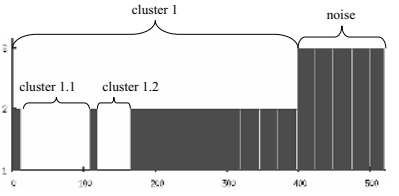


(b) Scalability w.r.t. dimensionality.

Fig. 2. Scalability of HiSC



(a) Data set.



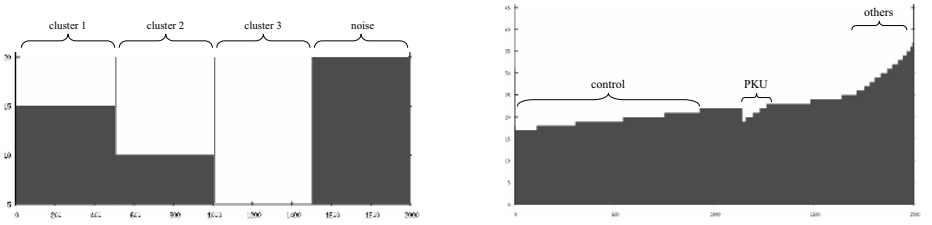
(b) Rechability diagram.

Fig. 3. Results on DS1 (3D)

The results of HiSC applied to DS1 are depicted in Figure 3(b). As it can be seen, the complete hierarchical clustering structure can be obtained from the resulting reachability plot. In particular, the nested clustering structure of the two 1D subspace clusters embedded within the 2D subspace cluster can be seen at first glance. Similar observations can be made when evaluating the reachability diagram obtained by HiSC on DS2 (cf. Figure 4(a)). HiSC has no problems with the three subspace clusters of considerably different dimensionality. The clusters can again be visually explored at first glance.

We also applied PreDeCon and PROCLUS on DS1 and DS2 for comparison. Neither PreDeCon nor PROCLUS are able to detect the hierarchies in DS1 and the subspace clusters of significantly different dimensionality.

We applied HiSC to a real-world data set containing metabolic screening data of 2,000 newborns. For each newborn, the blood-concentration of 43 different metabolites were measured. Thus, the data set is 43-dimensional containing 2,000 objects. The newborns are labeled by one of three categories. The healthy patients are marked by "control", newborns suffering phenylketonuria (a well-known metabolic disease) are labeled with "PKU", and newborns suffering any other metabolic disease are labeled with "others". The resulting reachability plot HiSC generates when applied to this data set is visualized in Figure 4(b). As it can be seen, HiSC produced a large hierarchy of 17D to 22D subspace clusters nested into each other. All these clusters contain approximately 98% newborns marked with "control". A second hierarchy of nested clusters contains only newborns marked with "PKU". The rest is a mix of all three categories.



(a) Results on DS2. (b) Results on metabolome data.

Fig. 4. Results on higher-dimensional data

4 Conclusions

In this paper, we presented HiSC, the first subspace clustering algorithm for detecting hierarchies of subspace clusters. HiSC scales linearly in the dimensionality of the data space and quadratically in the number of points. Several comparative experiments using synthetic and real data sets show that HiSC has a superior performance and effectivity compared to existing methods.

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. SIGMOD. (1998)
2. Cheng, C.H., Fu, A.W.C., Zhang, Y.: Entropy-based subspace clustering for mining numerical data. In: Proc. KDD. (1999)
3. Kailing, K., Kriegel, H.P., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: Proc. SDM. (2004)
4. Kriegel, H.P., Kröger, P., Renz, M., Wurst, S.: A generic framework for efficient subspace clustering of high-dimensional data. In: Proc. ICDM. (2005)
5. Aggarwal, C.C., Procopiuc, C.M., Wolf, J.L., Yu, P.S., Park, J.S.: Fast algorithms for projected clustering. In: Proc. SIGMOD. (1999)
6. Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.M.: A Monte Carlo algorithm for fast projective clustering. In: Proc. SIGMOD. (2002)
7. Böhm, C., Kailing, K., Kriegel, H.P., Kröger, P.: Density connected clustering with local subspace preferences. In: Proc. ICDM. (2004)
8. Sibson, R.: SLINK: An optimally efficient algorithm for the single-link cluster method. The Computer Journal **16** (1973) 30–34
9. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. SIGMOD. (1999)

Integrating Pattern Mining in Relational Databases

Toon Calders, Bart Goethals, and Adriana Prado

University of Antwerp, Belgium
{toon.calders, bart.goethals, adriana.prado}@ua.ac.be

Abstract. Almost a decade ago, Imielinski and Mannila introduced the notion of *Inductive Databases* to manage KDD applications just as DBMSs successfully manage business applications. The goal is to follow one of the key DBMS paradigms: building optimizing compilers for ad hoc queries. During the past decade, several researchers proposed extensions to the popular relational query language, SQL, in order to express such mining queries. In this paper, we propose a completely different and new approach, which extends the DBMS itself, not the query language, and integrates the mining algorithms into the database query optimizer. To this end, we introduce *virtual mining views*, which can be queried as if they were traditional relational tables (or views). Every time the database system accesses one of these virtual mining views, a mining algorithm is triggered to materialize all tuples needed to answer the query. We show how this can be done effectively for the popular association rule and frequent set mining problems.

1 Introduction

Almost a decade ago, Imielinski and Mannila [9] introduced the concept of an *Inductive Database*, in which a *Knowledge and Data Discovery Management System* (KDDMS) manages KDD applications just as DBMSs successfully manage business applications. Generally speaking, besides allowing the user to query the data, the KDDMS should also give users the ability to query patterns and models extracted from these data. In this context, several researchers proposed extensions to the popular relational query language, SQL, as a natural way to express such mining queries [8,10,11].

In our work, we aim at extending the DBMS itself, not the query language. That is, we propose an approach in which the user can query the collection of all possible patterns as if these are stored in relational tables. The main challenge is how this storage can be implemented effectively. After all, the amount of all possible patterns can be extremely large, and impractical to store. For example, in the concrete case of itemsets, an exponential number of itemsets would need to be stored. To resolve this problem, we propose to keep these pattern tables virtual. That is, as far as the user is concerned, all possible patterns are stored, but on the physical layer, no such complete tables exist. Instead, whenever the user queries such a pattern table, or *virtual mining view*, an efficient

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 454–461, 2006.
© Springer-Verlag Berlin Heidelberg 2006

data mining algorithm is triggered by the DBMS, which materializes at least those tuples needed to answer the query. Afterwards, the query can be executed as if the patterns had been there before. Of course, this assumes the user poses certain constraints in his query, asking for only a subset of all possible patterns, which should then be detected and exploited by the data mining algorithm. As a first step towards this goal, we propose such a constraint extraction procedure starting from a collection of simple constraints.

Notice that the user can now query mining results by using a standard relational query language, such as SQL. Furthermore, the user does not need to deal with the mining algorithms themselves as these are transparently triggered by the DBMS. We show how this approach can be implemented for the popular association rule and frequent set mining problems.

2 Related Work

The idea to integrate data mining into databases has been addressed in a number of works [8,9,10,11]. Some of these works focus on extensions of SQL, as a natural way to give the user the ability to mine the data. For example, the query language *DMQL* was proposed by Han *et. al.* for mining relational databases [8]. *DMQL* adopts an SQL-like syntax for mining different kinds of rules such as classification rules and association rules. Another example, is the *MINE RULE* operator proposed by Meo *et. al.* [11], designed as an extension of the SQL language. This operator was proposed specifically for association rule mining discovery. The *MSQL* language [10], proposed by Imielinski and Virmani, is also focussed on association rules. It extends SQL with the operators *GetRules* and *SelectRules* that can, respectively, generate and query a set of association rules. Other examples of query languages for data mining are $\mathcal{LDL}^{++}$ [13] and ATLaS [14] of Wang and Zaniolo, which are extensions of $\mathcal{LDL}$ and SQL, respectively. A more theoretical study of a data mining query language is the data mining algebra proposed by Calders *et. al.* [3].

3 Virtual Mining Views

An association rule, defined over a set of items $\mathcal{I}$,is an implication of the form $X \Rightarrow Y$, where $X, Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. We say that X is the antecedent and Y the consequent of the rule [2]. Let D be a collection of transactions, where each transaction is a subset of $\mathcal{I}$. Then, the rule $X \Rightarrow Y$ holds in D with support s, if s transactions in D contain $X \cup Y$, and confidence c if $c\%$ of the transactions in D that contain X also contain Y.

The transaction database D can be stored as a binary relational data table. For each transaction, there is a set of tuples of the form $(tid, item)$, where *tid* is the transaction identifier and *item* is an item in that transaction. Note that this table can also be implemented as a database view on the real data, as this is typically not represented in such a binary relation. Then, the association rules,

generated from the mining of D, can be represented in the same database where D is stored, by using the following schema.

1. *Sets*$(sid, item)$: This table represents all itemsets. A unique identifier *sid* is associated to each itemset and, *item* is an item in that itemset.
2. *Supports*$(sid, supp)$: This table represents the supports of the itemsets in the *Sets* table. For each itemset, there is a tuple where *sid* is the identifier of the itemset and *supp* the support.
3. *Rules*$(rid, sida, sidc, sid, conf)$: This table represents all association rules. For each rule $X \Rightarrow Y$, there is a tuple with an association rule identifier *rid*, the antecedent identifier *sida*, the consequent identifier *sidc*, the set identifier *sid* of the complete itemset $(X \cup Y)$, and *conf* the confidence.

Note that the choice of the schema for representing itemsets and association rules also implicitly determines the complexity of the queries a user needs to write. For example, one of the three set identifiers for an association rule, *sid*, *sida* or *sidc*, is redundant, as it can be determined from the other two. Nevertheless, it would also imply the user would have to write much more complicated queries. Essentially, only the data table D and the *Sets* table are necessary in order to be able to select itemsets or association rules in a single query. Indeed, without the *Sets* table, this is impossible because such a query explicitly generates the powerset of $\mathcal{I}$, which is well known to be impossible in SQL [1].

It is of course also still possible to add more attributes to these tables in which, for example, also other interestingness measures could be stored.

The main goal of the proposed pattern tables is to give the user the ability to query data mining results in the same way as traditional relational tables are queried. As already explained, however, it is intractable to store all $2^{|\mathcal{I}|}$ itemsets or $3^{|\mathcal{I}|}$ association rules. On the other hand, the entire set of patterns does not always need to be stored, but only the patterns that satisfy the constraints within the user's query (e.g., minimum support, minimum confidence, etc.). Therefore, in our proposal, the pattern tables are actually empty, and after the user has posed his query, the necessary patterns are materialized by the mining algorithm, immediately before the DBMS answers the query. Of course, this means that the DBMS should be able to detect the constraints in the given query. In our approach, we extract such constraints from a relational algebra expression equivalent to the user's query. Note that every SQL-query can easily be transformed into an equivalent relational algebra expression [7]. Such a relational algebra expression has the advantage of being procedural as compared to SQL, which is declarative, making the constraint extraction easier.

4 Extracting Constraints from Queries

For reasons of simplicity, we study a restricted class of constraints: for association rules we have minimal and maximal support, minimal and maximal confidence, plus the constraints that a certain item must be in the antecedent, in the consequent, in the antecedent or the consequent, and Boolean combinations of these.

The stricter the constraints we can extract, the more efficient query evaluation will become, because the number of tuples that needs to be materialized decreases when the extracted constraint is stricter. Note, however, that extracting the best possible constraint, i.e. the most strict, is theoretically impossible, even in the restricted case considered here. Indeed; suppose, for the sake of contradiction, that an algorithm exists that always extracts the best possible constraints. Then, this algorithm allows us to decide whether an SQL-query will always return an empty answer as follows: given a query q, run the assumed algorithm on the following query q': **select** *sid* **from** *Sets* **where** not exists (q); Obviously, on the one hand, if q always returns the empty answer, q' will never return any *sid*. In that case, the most strict constraint is "false"; i.e., no itemsets nor rules should be mined. On the other hand, if q is not empty, q' will produce the *sid* of every itemset, and hence, the constraint must be "true". Therefore, q is non-empty if and only if the assumed algorithm returns the constraint "true". Deciding non-emptiness of SQL-queries, however, is well-known to be undecidable [1]. Therefore, the algorithm proposed in this section is necessarily incomplete. For simple queries, however, it will find strict constraints, as will be illustrated with some examples.

As already explained earlier, the proposed constraint extraction algorithm does not work directly on the SQL-query, but on an equivalent relational algebra expression, which can be easily generated by existing algorithms.

Relational Algebra. A relational algebra expression is a sequence of set operations on the relations, resulting in the answer of the query. Consider, for example, the SQL-query shown in Fig. 1. The query asks for the rules and their confidences, that have the item *apple* in the antecedent or consequent of the rule, support of at least 40 and confidence of at least 80%. An equivalent relational algebra expression for this SQL-query is

$$\pi_{R.rid,R.conf}\ \sigma_{S_2.sid=R.sid}\ \big(\sigma_{S_1.sid=S_2.sid}\ (\sigma_{S_1.item=apple}\ \mathit{Sets}) \times (\sigma_{S_2.supp\geq 40}\ \mathit{Supports})\big) \times (\sigma_{R.conf\geq 80\%}\ \mathit{Rules})$$

In this expression, $\sigma_{R.conf\geq 80\%}\ \mathit{Rules}$ expresses that we only select those tuples from the relation *Rules* that satisfy the constraint $R.conf \geq 80\%$, $\times$ constructs the Cartesian product of two relations, i.e., for every tuple of the first relation, and every tuple of the second relation, a new tuple that is the concatenation of the two tuples is in the result relation of $\times$. $\pi_{R.rid,R.conf}$ produces a new relation that has only the attributes $R.rid$ and $R.conf$.

Notice that this expression can also be represented by its syntax tree, as is illustrated in Fig. 1. For the sake of simplicity, we will continue working with such expression trees instead of the relational algebra expressions themselves.

Algorithm. Given a query q as input, first, an equivalent tree of relational algebra is constructed. As every leaf node in this tree represents a table or a virtual mining view, the goal of the algorithm is to find which tuples should be present in those nodes representing a virtual mining view, in order to answer to

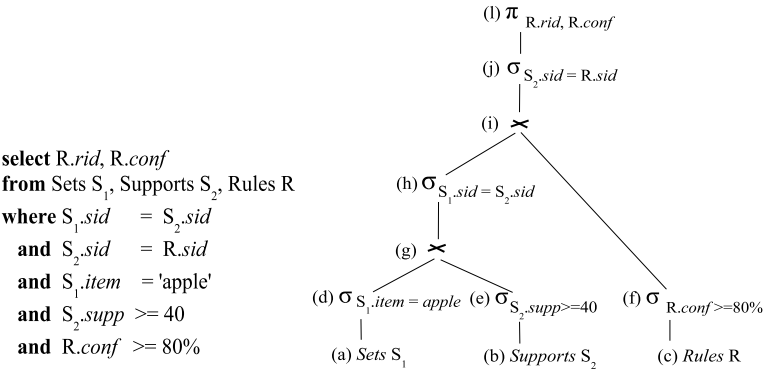


Fig. 1. An example query and its corresponding expression tree of relational algebra

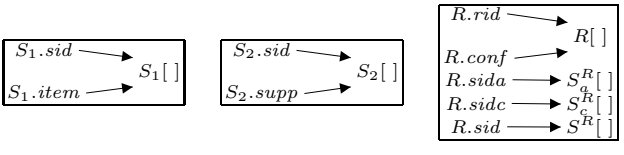
the query. Thus, actually, the algorithm determines, for each of the aforementioned nodes, a constraint. For example, in Fig. 1, there are three leaf nodes that represent virtual mining views: one with the *Sets* view, one with the *Supports* view, and one with the *Rules* view. The goal of the algorithm is then to identify that, in order to answer the query, it suffices to have in (a) and (b) only tuples that come from itemsets having the item *apple* and with support of at least 40, and in (c) only tuples that come from association rules with confidence of at least 80% and generated from the same collection of itemsets present in (a) and (b). We denote the subset of tuples of a virtual mining view V that come from itemsets that satisfy ϕ by $V[\phi]$. E.g., the subset of *Sets* needed to be materialized for node (a) can be denoted $Sets[apple \wedge supp \geq 40]$.

Notice that because leaf nodes always correspond to the tables or virtual mining views in the from-clause of the query, the algorithm finds, for every virtual mining view in the from-clause, a constraint. If one virtual mining view is used multiple times, multiple constraints will be extracted. This is not a problem, as we can easily combine these different constraints by taking the union.

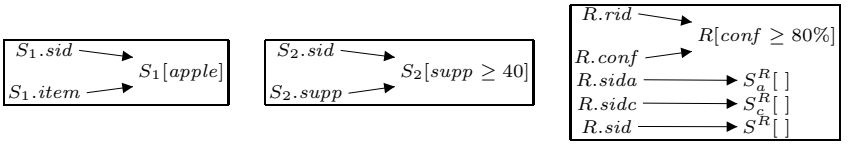
In this context, the procedure to extract the constraints on the virtual mining views is as follows. Starting from the leaves, going bottom-up, the algorithm determines for every node n in the expression tree which tuples should be in the virtual mining views in order to answer the *subquery* associated with that node, that is, the query represented by the subtree rooted at n. To answer the subqueries associated with the leaf nodes themselves, obviously, all tuples should be in, since the subquery is essentially asking for all tuples in the virtual mining view. Going up, however, it becomes clear that, in fact, not the complete virtual mining view is needed. E.g., in Fig. 1, for node (a), all tuples of *Sets* are needed. When going up to node (d), however, we see that only those tuples satisfying $(item = apple)$ are needed. Henceforth, to answer the subquery rooted at (d), it suffices that the virtual mining view *Sets* only contains those tuples that come from itemsets having the item *apple*, that is, only $Sets[apple]$ is needed.

In the computation of the constraints on the virtual mining views, for every node n, we annotate every node with a three-tuple $(\mathcal{A}, \mathcal{V}, \mathcal{O})$. In this three-tuple, $\mathcal{A}$ is the set of attributes $\{A_1, \ldots, A_n\}$ of that node, $\mathcal{V}$ is the set of virtual views with the constraints for n $\{V_1[\phi_1], V_2[\phi_2], \ldots, V_m[\phi_m]\}$, and $\mathcal{O}$ a set of pairs $A_i \longrightarrow V_j$ denoting that the values in attribute A_i originate from the view V_j. Notice that the values in an attribute can originate from more than one view at the same time, namely if two views have been joined on this attribute. We now give rules to compute the annotation of a node in the expression tree, if the annotations of its child nodes have been given. In this section we will restrict the detailed technical explanation to the constructions needed for the example query given in Fig. 1. For the complete explanation, we refer the reader to [4].

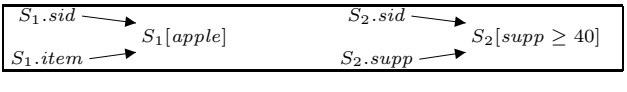
Leaf annotation. The annotations for the leaves *Sets* S_1 (node (a)) and *Supports* S_2 (node (b)) denote that there are two attributes connected to the views, named S_1 and S_2, respectively, both associated to an empty constraint ([ ]). For the leaf *Rules* R (node (c)), the annotation is a bit more complicated, as there are in fact four objects that can be constrained: the antecedent of the rule, the consequent, the union of the two, and the rule itself. Therefore, four variables are introduced that represent respectively these objects: S_a^R, S_c^R, S^R, and R. Thus, the annotations for the leaves (a), (b) and (c) are, respectively:



Annotation of internal nodes. In node (d), only those tuples coming from (a) that satisfy the condition (*item* = *apple*) are selected. From the annotation of node (a), we observe that the attribute $S_1.item$ originates from the view S_1 (*Sets*). Thus, we can actually associate the constraint [*apple*] to S_1. By similar reasoning, we can associate the constraint [*supp* ≥ 40] to S_2 and the constraint [*conf* $\geq 80\%$] to R. The annotation for nodes (d),(e) and (f) are, respectively:

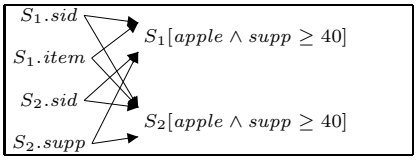


In node (g), a cartesian product is made from the tuples coming from nodes (d) and (e). Therefore, all tuples from nodes (d) and from node (e) should be considered at this node. The annotation for node (g) is then the union of the annotations of its child nodes:



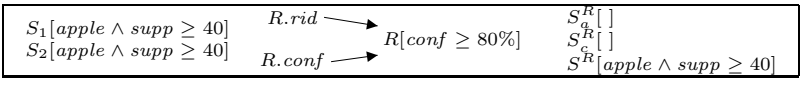
In node (h), only those tuples coming from node (g) that satisfy the condition ($S_1.sid = S_2.sid$) are selected. From the annotation of node (g), we can see that

$S_1.sid$ is connected to the view S_1 with constraint $[apple]$ and $S_2.sid$ is connected to the view S_2 with constraint $[supp \geq 40]$. Then, according to the related condition, the tuples really needed to be considered in node (h) are those coming from the itemsets present in both of the views $S_1[apple]$ and $S_2[supp \geq 40]$, that is, the same collection of itemsets having the item *apple* and with support greater than or equal to 40. We can thus associate the constraint $[apple \wedge supp \geq 40]$ to both views. Furthermore, as the itemsets come from the same collection, the set of attributes of S_1 are now connected to the view S_2 and vice versa. The annotation for node (h) is as follows:



The annotation for nodes (i) and (j) are constructed in the same way as the annotations for nodes (g) and (h), respectively.

Finally, root node (l) projects the tuples coming from node (j) over the attributes $R.rid$ and $R.conf$. Its annotation is similar to that one of its child node (j), keeping, however, only the attributes $R.rid$ and $R.conf$ and its connections. The annotation for this node is then:



Observe that the annotation of the root node (l) has all the necessary information we need in order to know exactly which tuples should be materialized by the DBMS: According to the constraint $[apple \wedge supp \geq 40]$ associated to the views S_1 and S_2, we can identify that the views *Sets* and *Supports* should contain all tuples coming from itemsets having the item *apple* and with support of at least 40. According to the constraint associated to the view R, it is easy to deduce that the view *Rules* should contain the tuples coming from association rules with confidence of at least 80%. Moreover, as S^R is also associated to the constraint $[apple \wedge supp \geq 40]$, the association rules should be generated from the same collection of itemsets present in the views *Sets* and *Supports*. Note that the constraints associated to S_a^R and S_c^R are both empty, which means that there are no constraints considering the presence of items in the antecedent nor the consequent of the rules, respectively. With this information, the DBMS can trigger the necessary data mining algorithms with the identified constraints.

5 Conclusion

This paper proposes a different and new approach in which association rule mining results can be queried as if they were stored in traditional relational tables. This approach is based on the existence of *virtual mining views* that

represent mining results. Every time the user queries one of these views, data mining algorithms are triggered by the DBMS in order to materialize, according to the constraints within the given query, the patterns needed to answer it.

From the user's point of view, the virtual mining views will always contain all association rules and itemsets, but, according to our proposal, none of the patterns should be actually stored. In reality, the action of querying a virtual mining view triggers a data mining algorithm or a set of data mining algorithms transparently to the user, which means that the user does not need to know how to use data mining algorithms. Moreover, due to the fact that only the DBMS is extended, the user can query the data by using a standard relational query language. Extensions of query languages are not necessary in our approach.

Acknowledgement. This work was partially funded by the EU project "IQ", and the FWO project "Foundations for Inductive Databases". The second author would also like to thank Jan Van den Bussche for many interesting discussions and suggestions resulting in several of the ideas presented in this paper.

References

1. S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases*, Addison-Wesley (1995).
2. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *20th VLDB Conference* (1994) 487–489.
3. T. Calders, L. V.S. Lakshmanan, R. T. Ng and J. Paredaens. Expressive Power of an Algebra for Data Mining. *Manuscript* (2006).
4. T. Calders, B. Goethals and A. Prado. Constraint Extraction from SQL-queries. *Manuscript* (2006).
5. B. Goethals, J. Van den Bussche and K. Vanhoof. Decision Support Queries for the Interpretation of Data Mining Results. *Manuscript* (1998).
6. B. Goethals and J. Van den Bussche On Supporting Interactive Association Rule Mining. In *2nd DaWaK* (2000) 307–316.
7. H. Garcia-Molina, J. D. Ullman and J. Widom. Database System Implementation. Prentice-Hall, Inc. (2000).
8. J. Han, Y. Fu, W. Wang, K. Koperski and O. Zaiane. DMQL: A Data Mining Query Language for Relational Databases. In *SIGMOD DMKD Workshop* (1996).
9. T. Imielinski and H. Mannila. A Database Perspective on Knowledge Discovery. *Communications of the ACM*, Vol. 39 (1996) 58–64.
10. T. Imielinski and A. Virmani. MSQL: A Query Language for Database Mining. *Data Mining and Knowledge Discovery*, Vol. 3 (1999) 373–408.
11. R. Meo, G. Psaila and S. Ceri. An Extension to SQL for Mining Association Rules. *Data Mining and Knowledge Discovery*, Vol. 2 (1998) 195-224.
12. T. Mitchell Machine Learning. McGraw Hill (1997).
13. H. Wang and C. Zaniolo. Nonmonotonic Reasoning in LDL++. In *Logic-Based Artificial Intelligence*, J. Minker, Ed. Kluwer Academic Publishers (2000) 523–544.
14. H. Wang and C. Zaniolo. ATLaS: A Native Extension of SQL for Data Mining. In *3rd SIAM Conference* (2003) 130–144.

Discovering Patterns in Real-Valued Time Series

Joe Catalano, Tom Armstrong, and Tim Oates

University of Maryland Baltimore County
Baltimore, MD 21250 USA
{jcat1, arm1, oates}@umbc.edu

Abstract. This paper describes an algorithm for discovering variable length patterns in real-valued time series. In contrast to most existing pattern discovery algorithms, ours does not first discretize the data, runs in linear time, and requires constant memory. These properties are obtained by sampling the data stream rather than processing all of the data. Empirical results show that the algorithm performs well on both synthetic and real data when compared to an exhaustive algorithm.

1 Introduction

Many of the data generated and stored by science, government, and industry are multi-variate, real-valued, and streaming. These time series data come from such diverse sources as financial markets, climatological satellites, and medical devices. The potential uses of time series data are as varied as their sources. In some cases, the goal is to make accurate predictions (e.g., finding patterns in the fluctuations of the price of one stock that predict the price of another stock 3 days hence). In other cases, the goal is to gain a deeper understanding of the underlying system generating the data. This paper is concerned with the latter task, and presents a novel algorithm for finding recurring patterns (sometimes called motifs [1]) in time series.

Most algorithms for discovering patterns in time series have one or more of the following characteristics. The most common characteristic is an inability to work with real-valued data except through prior discretization [2,3]. Even in those cases where real-valued data are acceptable, multi-variate data typically are not [4]. The algorithms also tend to be batch [5], rather than incremental, which poses problems when the datasets are large or come from a high-volume stream. Finally, there are often assumptions about the number or temporal extent of patterns that exist in the data [6]. In contrast, we present an incremental algorithm with linear time and constant memory requirements for finding recurring patterns in real-valued, multi-variate streaming data wherein the number and temporal extent of the patterns are not known in advance.

The algorithm obtains these desirable properties by sampling. Given a data stream, a user-specified number of large *candidate windows* of data are sampled from the stream, sub-windowed, and stored. As time progresses, *comparison windows* of the same size are periodically sampled, sub-windowed, compared to all of the sub-windows from each of the candidate windows, and finally discarded.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 462–469, 2006.
© Springer-Verlag Berlin Heidelberg 2006

The similarity scores (according to DTW) of the k most similar comparison sub-windows are kept for each candidate sub-window. If a candidate window contains an instance of a pattern, then some of its sub-windows will contain parts of that pattern, and the sampling process will yield comparison windows and sub-windows with the same properties. To distinguish patterns from noise, the mean similarity of the best matches for each candidate sub-window is compared to a distribution of mean similarities constructed so as to enforce the null hypothesis that the matching sub-windows do not contain instances of the same pattern. When the null hypothesis is rejected, the candidate sub-window probably contains a part of a pattern instance, and adjacent sub-windows with this property are "stitched" together to obtain the entire pattern.

Empirical results with a variety of datasets are reported. First, synthetic datasets in which known patterns are added to a background of noise are used to explore the ability of the algorithm to discover patterns of varying lengths and frequencies of occurrence in the face of changes to user-defined parameters. Second, the algorithm is run on real datasets for which the "true" patterns are not known. The discovered patterns are compared to those found by an exceptionally expensive algorithm that performs an exhaustive search for patterns of all lengths in the data. Results show that our algorithm finds many of the same patterns found by the exhaustive algorithm, but with limited computation and memory.

The remainder of this paper is organized as follows. Section 2 describes approaches to discovering patterns in data through discretization. Section 3 presents our sampling algorithm and complexity analysis. Section 4 contains empirical results of our algorithm on real-valued data. Finally, section 5 summarizes our contribution and points to future research directions.

2 Related Work

Extensive work has been done in the area of time series data mining, but little of it has focused on mining recurring patterns in real-valued time series. Some have applied clustering techniques to time series to mine interesting features of the data.

Dynamic Time Warping (DTW) is used in Oates et al. [7] to cluster the experiences of a mobile robot, using robotic sensor data as the source of the time series, and in [8] to cluster multi-variate real-valued time series produced by selecting one of k HMMs. While not focused on pattern discovery, they establish that DTW can reliably be used as a similarity measure of real-valued multivariate data.

Colomer et al. [2] use DTW to classify patterns belonging to the same class of operating situations in a level control system. Unlike our approach, they apply DTW to *episodes* rather than to the original time series. Keogh and Pazzani [3] propose a modification to DTW that operates on a piecewise linear representation of the data. Again this differs from our approach as it does not operate on the raw observations.

Lin et al. [4] define a *motif* as a previously unknown frequently occurring pattern, and introduce a discovery method that uses a modified Euclidean distance function to improve the complexity of the search. To achieve this performance they must reduce the dimensionality of the time series by means of a piecewise aggregate approximation and then further transform the time series into a discrete representation. Chiu et al. [1] extend Lin's work, addressing the scalability of the motif discovery algorithm and its ability to discover motifs when noise is present in the time series. Lin et al. [9] extend the symbolic representation introduced in [4] to better work with algorithms for streaming time series.

Finally, Oates [5] investigates a variation of the pattern discovery problem, wherein they determine what makes a set of sequences different from other sequences obtained from the same source. The approach operates on multi-variate real-valued time series and uses DTW as the similarity measure, but does not use a sampling approach.

3 Algorithm

Our sampling algorithm, figure 1, accomplishes two main tasks; the first is to discover windows from a time series that contain pattern instances and the second is to remove noise prefixes and suffixes in windows containing patterns. It accomplishes these goals by repeatedly sampling fixed windows of the time series looking for pairs of windows that have a small distance under DTW. We use the distance between the windows as a measure of similarity, with large distances indicating dissimilarity. We further constrain the algorithm by requiring it to discover patterns without a priori knowledge of their existence. We must then define some threshold for rejecting a match when the DTW distance between the windows is too large. Since the goal is to distinguish patterns from noise, a distribution of DTW distances between noisy windows is computed and used as a reference for thresholding the quality of a match. The algorithm performs these tasks using bounded time and memory by fixing the number and size of the windows that are compared. For an incremental version we sample windows on demand.

3.1 Noise Distribution

The sampling algorithm has no a priori knowledge of the existence of patterns in the time series. We define the null hypothesis to be that two randomly sampled windows do not contain instances of the same pattern. A distribution of window similarities must be computed as a basis for rejecting the null hypothesis when two windows contain a pattern instance. This is problematic because the core assumption is that given a large enough window of time series, it will contain some or all of a pattern instance with high probability if the patterns occur frequently. We create windows containing non-contiguous observations from the time series which ensures that these windows contain a pattern instance with low probability. That is, we create a noise window of length w by randomly sampling

and concatenating w individual observations. Warping these *noise windows* with normal windows gives us a null hypothesis distribution.

3.2 Pattern Discovery

In the pattern discovery phase of the algorithm, we repeatedly sample *candidate windows* from the time series and store the k-best matching *comparison windows* to each. Our goal is to identify frequently occurring patterns, thus we rely on sampling a sufficient number of windows to increase the probability of capturing multiple instances of a pattern. For the algorithm to be successful, the window length we choose must be large enough to fully contain any pattern we expect to find, but this also means the windows will contain noise as well. Having noise in the windows will increase the distance between them making it difficult to identify legitimate patterns. This is in addition to the problem of extracting only the pattern from the window. To address both of these problems we consider *sub-windows*. The relatively small size of a sub-window makes it useful for addressing this issue of granularity. Large windows contain noise and a large ratio of noise to pattern will yield poor results under any distance measure because the percentage of the window that will not have a strong matching region in another window will be great.

The patterns are discovered as follows. Create two sets of sub-windows, the *candidate set*, denoted *candSW*, and the *comparison set*, denoted *compSW*. It is the candidate set from which we reject sub-windows that come from the noise distribution and identify patterns from the remaining sub-windows. To populate these sets, sample a window W having length w and generate all sub-windows W_i having length $\bar{w}$. This yields $(w - \bar{w}) + 1$ sub-windows which are added to either set (*candSW* or *compSW*) and repeat this process to a specified bound. Normalize each sub-window to have mean 0 and standard deviation 1. When both sets are populated, apply DTW to all pairs of sub-windows between the two sets. Group the sub-windows in *compSW* by the window from which they were created and add only the best W_i in *compSW* from each group to the list of matches for W_i to which it is being compared. After warping all pairs, reduce the list of matches for each W_i in *candSW* to the k with the smallest distance measures.

In the final step of the algorithm, bad matches are removed from the candidate set. Recall the noise distribution that was created by warping normal sub-windows to sub-windows containing pure noise. As with candidate sub-windows, keep only the k-best matches for each noise sub-window. Sort the noise set by increasing average distance and a rejection threshold, γ, is established. Given some α, $\gamma = \lfloor(\alpha * n)\rfloor^{th}$ average warp distance where n is the number of noise sub-windows. Reject those sub-windows that have an average warp distance greater than γ on the basis that the observed value is common in the noise distribution and therefore is not likely to be a pattern instance. After removing the sub-windows containing bad matches, repeat the entire process using the now smaller candidate set, the same noise set, and a new comparison set. The process of eliminating candidate sub-windows is inherently error-prone because the comparison windows are randomly chosen and therefore we may not get enough

windows containing patterns to accurately accept or reject the candidate sub-windows. By running multiple iterations of the algorithm we reduce the amount of error introduced into the process.

```
SAMPLING(timeSeries, w, w̄, iterations, alpha)
 1  amtToSample ← AMOUNTTOSAMPLE(timeSeries, w, 50)
 2  compSW ← CREATESWSET(timeSeries, w, w̄, amtToSample)
 3  candSW ← CREATESWSET(timeSeries, w, w̄, amtToSample)
 4  noiseSW ← CREATENONCONTIGUOUSSWSET(timeSeries, w)
 5  combSW ← {∅}
 6  ADDALL(noiseSW, combSW)
 7  ADDALL(candSW, combSW)
 8  for i ← 1 to iterations
 9       do
10          if (iterations > 1)
11            then
12                 compSW ← CREATESWSET(timeSeries, w, w̄, amtToSample)
13                 COMPAREALLSUBWINDOWPAIRS(candSW, compSW)
14            else
15                 COMPAREALLSUBWINDOWPAIRS(combSW, compSW)
16
17          REMOVEREJECTS(alpha, candSW, noiseSW)
18
```

Fig. 1. Batch Mode Pattern Discovery

When all iterations have completed we then stitch together the remaining candidate sub-windows to form patterns. A pattern is formed by combining all overlapping and adjacent sub-windows. The resulting window is then considered to be a complete pattern instance.

The number of subwindows and the number of comparisons by DTW dominates the space and time complexity, respectively. DTW has quadratic time and space complexity, but the algorithm only considers pairs of subwindows of length $\bar{w}$, resulting in constant time and space costs. The algorithm samples n candidate windows of length w and m comparison windows of length w requiring $\mathrm{O}(n)$ and $\mathrm{O}(m)$ time and space. Each window of length w has $(w - \bar{w} + 1)$ subwindows of length $\bar{w}$. Therefore, there are a total of $n * m * (w - \bar{w} + 1)^2$ subwindows to compare and store, or $\mathrm{O}(nm)$ time and space. When considering the incremental version of the algorithm, $m = 1$ so the complexity is linear in the number of candidate windows.

4 Experiments

We have evaluated our algorithm by studying performance on a synthesized time series with a recurring, embedded noisy pattern. Then we explore the uni-variate time series of the Standard & Poor's 500.

We generated a uni-variate time series of 10,000 observations sampled uniformly in the range $[0, 10]$. The first 18 observations were used as a template for the pattern to be embedded in the data. The template is duplicated 49 times and noise chosen uniformly in the range $[-1.0, 1.0]$ is added to each observation in every copy of the template. We then insert the copies into the time series starting at position 100 and incrementing by 100 until all 49 have been embedded.

We ran the sampling algorithm on the synthetic data varying the values for $\bar{w}$ and α and we fixed w as $10*\bar{w}$. To $\bar{w}$ we assigned values from the set $\{5, 10, 15, 20\}$ and to α values from the set $\{0.05, 0.10, 0.15, 0.20\}$. It is expected that the noise distributions will be normally distributed, and the false-positive rate should increase proportionately with greater values of α. Sub-windows that overlap a pattern should have higher quality matches than those that do not overlap a pattern, and when sub-window stitching is performed the stitched pattern should be 18 observations in length and align with the start of an embedded pattern. Running multiple iterations of the sampling algorithm for a given set of parameters, α and $\bar{w}$, should reduce the number of errors made when selecting candidate sub-windows to keep.

Type I and Type II errors made by the algorithm when it evaluates a sub-window for either acceptance or rejection are important statistical measures when determining if the algorithm is performing correctly. For our sampling algorithm, a Type I error is made when the *candidate* sub-window spans an instance of the embedded pattern but is rejected. Likewise, a Type II error is made when the *candidate* sub-window does not span an instance of the embedded pattern but is accepted. In this experiment we know the locations of every instance of the pattern. Therefore counting the exact number of Type I and Type II errors is possible.

Table 1. Type I & II Errors, $\bar{w} = 5$, $\alpha = 0.05$

	Iteration	Total	Accept	Error Rate
Pattern	1	209	105	49.7%
Pattern	10	43	42	2.3%
Pattern	25	27	26	3.7%
Not Pattern	1	1057	45	4.2%
Not Pattern	10	5	4	80%
Not Pattern	25	2	2	100%

Table 1 displays the totals, categorized as sub-windows that span only pattern observations and sub-windows that span only noise observations, after one, ten, and twenty five iterations of the sampling algorithm for $\bar{w} = 5$ and $\alpha = 0.05$. The data are a good indication that our noise distribution is an accurate model of the noise contained in the time series and performs well at eliminating sub-windows spanning noise. After 25 iterations, we have nearly eliminated all sub-windows spanning noise while keeping the percentage of Type I errors to a minimum. For different values of α (i.e. 0.10 and 0.20), the algorithm performed similarly.

4.1 Standard and Poor's 500

In our real dataset experiment, we ran our algorithm on the daily closing price of the S&P 500 for approximately 33 years. The data were 8760 uni-variate time series observations, each fell in the range $[4.4, 49.74]$. As was the case with the synthetic experiments, the parameters for this experiment were $\alpha = \{0.05, 0.10, 0.15, 0.20\}$ and $\bar{w} = \{5, 10, 15, 20\}$.

Unlike in the previous experiment, we did not have knowledge of any patterns occurring in this data. This made it necessary to perform an exhaustive search of the time series to produce a basis for evaluating the quality of the patterns discovered by our algorithm. Likewise, it is impossible to collect statistics with regard to the quantity of Type I and II errors that were made, because there is no way to determine when a window contains pattern observations and when it does not.

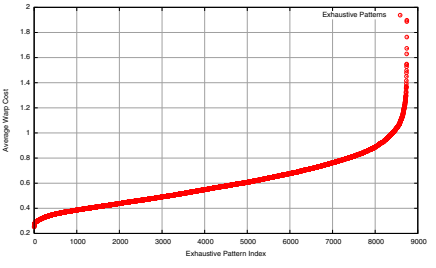


Fig. 2. S&P 500 Exhaustive Costs $\bar{w} = 10$

Figure 2 shows the patterns found exhaustively as a function of their average warp cost and sorted from best to worst. The exhaustive search warped all pairs of sub-windows for a given sub-window length (ignoring overlapping sub-windows) and maintained a list of the ten best matches for each candidate. The warp cost depicted in the figures are the average of those ten matches for each candidate sub-window. The plot shows that a small number of the candidates had exceptionally good matches and a small number had exceptionally poor matches.

The top-ten candidate sub-windows found using the sampling algorithm for sub-window size 10 all were in the 93rd-percentile out of 8751 sub-windows under the exhaustive search results. Three of the ten results were in the top-100 candidates in the exhaustive search results. We have experimented on additional datasets, including commonly cited multi-variate datasets (e.g. winding, subcutaneous). The results of our algorithm on these data are analogous to the uni-variate cases.

5 Conclusion

This paper described an incremental algorithm with linear time and constant memory requirements for finding recurring patterns in real-valued, multi-variate

streaming data wherein the number and temporal extent of the patterns is not known in advance. Empirical results with synthetic data showed that it successfully finds known patterns and is robust with respect to the settings of user-specified parameters. Empirical results with real data show that the patterns found by the algorithm compare favorably to an (impractically expensive) exhaustive algorithm. Future work will focus on applications with high-volume data streams (e.g., audio data) and automated representation tuning by, for example, searching over sets of basis functions for those that yield high quality (low match cost) patterns.

References

1. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: 9th International Conference on Knowledge Discovery and Data Mining (SIGKDD'03). (2003) 493–498
2. Colomer, J., Melendez, J., Gamero, F.I.: Pattern recognition based on episodes and DTW, applications to diagnosis of a level control system. In: Proceedings of the Sixteenth International Workshop on Qualitative Reasoning. (2002)
3. Keogh, E., Pazzani, M.: Scaling up dynamic time warping to massive datasets. In Zytkow, J.M., Rauch, J., eds.: 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99). Volume 1704., Prague, Czech Republic, Springer (1999) 1–11
4. Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: Proceedings of the Second Workshop on Temporal Data Mining, Edmonton, Alberta, Canada (2002)
5. Oates, T.: Identifying distinctive subsequences in multivariate time series by clustering. In Chaudhuri, S., Madigan, D., eds.: Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, ACM Press (1999) 322–326
6. Bozkaya, T., Yazdani, N., Ozsoyoglu, Z.M.: Matching and indexing sequences of different lengths. In: CIKM. (1997) 128–135
7. Oates, T., Schmill, M.D., Cohen, P.R.: A method for clustering the experiences of a mobile robot that accords with human judgments. In: AAAI/IAAI. (2000) 846–851
8. Oates, T., Firoiu, L., Cohen, P.R.: Clustering time series with hidden markov models and dynamic time warping (1999)
9. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the Eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, San Diego, CA, USA (2002)

Classification of Dementia Types from Cognitive Profiles Data

Giorgio Corani[1], Chris Edgar[2], Isabelle Marshall[2],
Keith Wesnes[2], and Marco Zaffalon[1]

[1] IDSIA (Istituto Dalle Molle di Studi sull'Intelligenza Artificiale)
Manno, Switzerland
{giorgio, zaffalon}@idsia.ch
[2] Cognitive Drug Research Ltd
Goring-On-Thames, U.K.
{chrise, isabellem, keithw}@cognitivedrugresearch.com

Abstract. The Cognitive Drug Research (CDR) system is specifically validated for dementia assessment; it consists of a series of computerized tests, which assess the cognitive faculties of the patient to derive a *cognitive profile*. We use six different classification algorithms to classify clinically diagnosed diseases from their cognitive profiles. Good accuracy was obtained in separating patients affected by Parkinson's disease from demented patients, and in discriminating between Alzheimer's disease and Vascular Dementia. However, in discriminating between Parkinson disease with dementia (PDD) and dementia with Lewy bodies (DLB), the accuracy was only slightly superior to chance; the existence of a significant difference in the cognitive profiles of DLB and PDD is indeed questioned in the medical literature.

Keywords: CDR computerized assessment system, dementia, classification.

1 Introduction

Dementia is one of the most common disorders among the elderly; it causes a progressive decline in cognitive functions such as memory, attention and language. The Cognitive Drug Research (CDR) system [1] is widely used in clinical trials and has been specifically validated for use in dementia; it consists of a series of computerized tests (*tasks*), which assess some cognitive faculties of the patient, such as memory, attention, reaction times. The set of the measures collected during all tasks represents the *cognitive profile* of the patient.

In [3] the cognitive profiles returned by the CDR test are used to address two different classification problems: (a) to discriminate between demented patients and controls, and (b) to discriminate from among the different types of dementia. An accuracy higher than 90% was obtained on both tasks by using the Naive Credal Classifier, a generalization of the Naive Bayes Classifier to imprecise probabilities, or credal sets.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 470–477, 2006.
© Springer-Verlag Berlin Heidelberg 2006

In this paper, we propose a similar approach, but a few important differences in the data: (i) the number of tasks selected from the CDR battery is smaller, and mainly restricted to the attentional measures. Indeed, it is not clearly stated in literature whether the attentional measures of the cognitive profile are really different between some kinds of dementia, and therefore the subject is worthy of investigation. Moreover, a visit including only attentional tasks would take no more than 10 mins., while a complete administration of the CDR tasks would take between 30 and 45 mins. (ii) Standard deviation and number of outliers of each featured measure are available, while only the median was available in the previous study; indeed, fluctuations of cognitive faculties, captured by the standard deviation of the variables, are important to characterize the cognitive profiles, as shown in [4]; (iii) an enlarged set of dementias is considered, and Parkinson's patients are used instead of healthy controls to assess whether the system is able to discriminate between motor impairment and dementia. (iv) A peculiar investigation of this study is moreover the inter-comparison of the accuracies obtained using the cognitive profiles assessed at the first visit and at the third visit on the CDR test. Indeed, although patients are usually trained twice in the clinical practice on the tests prior to the definitive assessment of the cognitive profile at the third visit, performing the classification directly on the first-visit data could allow for time and money savings. We experimentally check whether first-visit data leads to classification accuracy better or statistically not different from third-visit data (hypothesis H_0). If H_0 is verified, we can indeed easily recommend the use of first-visit data. If on the contrary the experiment show a statistical improvement of the accuracy using third-visit data (hypothesis H_1), the judgment becomes more complex.

The aim of this paper is to better understand the domain of dementia analysis through cognitive profiles, which has been only rarely explored up to now by means of ML techniques, and to provide findings useful to ML scientists that will in the future work on similar data. We have looked for *robust* experimental findings, supported by the results of a set of classification approaches, rather than fine-tuning for performance a specific algorithm. We have therefore considered a set of different classification algorithms, including very well-known approaches such as J4.8, nave Bayes, logistic regression and other algorithms which performed well on our data (classification via regression, lazy, etc.). Further classification approaches have been excluded from the analysis because their performance was remarkably worse compared to that of the algorithms eventually selected.

2 The CDR Test

Currently, there are more than 30 studies referenced in dementia literature based on the CDR system. The system is entirely computerized, thus allowing for precise measurement of the latency of each response.

The following tasks are considered in this study:

- *Simple reaction time (SRT):* the patient should press the "yes" button as quickly as possible as the word "yes" is displayed on the monitor. The task is repeated 30 times.
- *Digit VIGilance task (VIG)*: a random target digit is constantly displayed on the monitor screen. A series of digits are then presented and the patient should press "yes" as quickly as possible as the digit in the series matches the target digit.
- *Choice reaction time* (*CRT*): either the word "no" or the word "yes" is displayed and the patient should press the corresponding button as quickly as possible. 30 trials are performed.
- *Delayed PICture recognition (DPIC):* a series of 14 pictures is presented on the monitor for the patient to remember. Afterwards, the same pictures are presented to the patient, together with 14 distracting pictures; for each picture the patient had to indicate whether or not it belongs to the first series.

For each task, several index of performances are recorded.

3 The Dataset

Two separate datasets contain the cognitive profiles assessed at the first and at the third visit on the CDR tasks. The dataset of the first visits contained 1842 records of cognitive profiles, while the dataset of the third visit contained 1670 records. Data were taken from patients before they entered different clinical trials. The datasets used in this study contains patients from Western Europe, Eastern Europe and Asia.

Different kinds of dementia are present in the dataset: Alzheimer disease (AD), Dementia with Lewy Bodies (DLB), Parkinson Disease with Dementia (PDD), Vascular Dementia (VAD); moreover, the dataset comprised patients affected by Parkinson's Disease (PD). PD patients are actually *not* demented, though they suffer a significant motor impairment. The distribution of the diseases is as follows: {AD 21%; DLB 10%; PDD 28.5%;VAD 34.5%; PD 6%} and it is almost identical for the datasets of the first and of the third visit.

In particular, there is not yet gold standard for the clinical distinction of AD and VAD, although there is the need of distinguishing between them because of the differences in the necessary treatments.

PDD and DLB are also very similar diseases: they are characterized by both dementia and parkinsonism, and their cognitive profiles have been found to be striking similar [4].

Therefore, three classification tasks appear as of scientific interest:

- *task 1*: to classify patients into three macro-classes as (AD-VAD, PDD-DLB, PD);
- *task 2a*: to discriminate between PDD and DLB;
- *task 2b*: to discriminate between AD or VAD. This is the only task for which features related to Delayed PIcture Recognition are available.

The tasks have been implemented separately from each other, and evaluated separately in this paper. However, in a real clinical use, they could be implemented within an actual cascade classification system, the output of classifier 1 possibly feeding, depending on the classification output, classifier 2a or 2b. Such an architecture would allow to easily use the additive DPIC features for tasks 2b.

4 Experimental Setting

Datasets have been analyzed using six different classification algorithms, implemented within the open source[1] WEKA software [5]. Classification algorithms have been used with their default settings.

We relied on the indication of the CDR staff as for the set of features to be used. All the considered features are numerical; however, we discretized them through the MDL-based supervised discretization algorithm originally proposed in [6]. Indeed, the experimental investigation carried out in [7] showed that quite frequently the use of such discretization algorithm leads to improved accuracy compared to the raw data. This turned out to be the case even for our dataset; for instance, J4.8 improved of about 2 accuracy points thanks to the use of discretized data.

The accuracy of the classifiers has been assessed via 10 runs of 10-fold cross validation. The statistical significances of the differences in accuracy have been tested via a t-test (5% significance); in particular, to properly manage the cross-validation errors, we used the corrected resampled t-test implemented in WEKA.

For tasks (1) and (2a) the features related to simple reaction time, choice reaction time, digit vigilance have been used; for task (2b), also the features related to picture recognition have been used.

5 Results

5.1 Classification Task 1: {(AD/VAD), (PDD/DLB), (PD)}

The accuracies obtained on this task are shown in Table 1. Depending on the classification algorithm, the classification accuracy ranges between 75% and 80%.

None of the 7 classification approaches showed a significant difference of accuracy working on the data of the first visit rather that on the data of the third visit; therefore, these results clearly support H_0.

Classification-via-regression and logistic regression appear as the best performing approaches. However, the objective of this study was mainly to get an indication about the obtainable accuracy, rather than fine tuning the algorithms for the best performance.

The confusion matrix for Logistic Regression is reported in Table 2; it shows that most misclassifications occurs between (AD-VAD) and (PDD-DLB), while it

[1] Available at the URL: http://www.cs.waikato.ac.nz/~ml/index.html

Table 1. Accuracy of different classification algorithms in task 1

	1st visit		**3d visit**		*Significant*
	average	*std. dev.*	*average*	*std. dev.*	*difference?*
classifier					
NAVE BAYES	75.81	3.18	74.90	3.46	NO
BAYES NETWORK	75.74	3.13	74.82	3.40	NO
J4.8 TREE	78.46	3.06	77.49	3.04	NO
SMO	78.12	2.89	78.11	2.90	NO
LOGISTIC REGRESSION	80.09	3.09	78.82	2.89	NO
LAZY.LBR	78.30	2.94	77.17	3.40	NO
CLASS. VIA REGR. (M5)	79.93	2.81	78.51	2.98	NO

Table 2. Confusion matrix for Logistic Regression on task 1

	AD-VAD	*DLB-PDD*	*PD* ←*classified as:*
AD-VAD	898	117	6
DLB-PDD	181	510	13
PD	20	30	67

is quite rare for Parkinson's disease to be confused with dementia. In particular, a demented patient was likely to be diagnosed as PD with almost negligible probability, while with slightly higher probability a PD patient is diagnosed as demented. However, this was probably due also to the very low proportion of PD in our dataset (6%).

5.2 Classification Task 2a: {(PDD), (DLB)}

By running the classifier on datasets containing all the instances of PDD and DLB patients for first and third visit, we measured an accuracy between 70% and 76%.

Table 3. Accuracy of different classification algorithms in discriminating between PDD and DLB on a *balanced* dataset

	1st visit		**3d visit**		*Significant*
	average	*std. dev.*	*average*	*std. dev.*	*difference?*
classifier					
NAVE BAYES	56.00	6.51	58.29	7.03	NO
BAYES NETWORK	56.08	6.34	58.32	7.02	NO
J4.8 TREE	54.73	6.24	57.68	7.28	NO
SMO	55.63	6.19	57.23	6.90	NO
LOGISTIC REGRESSION	55.91	6.50	58.00	7.26	NO
LAZY.LBR	55.92	6.65	58.29	7.03	NO
CLASS. VIA REGR. (M5)	55.77	6.64	56.58	7.11	NO

However, considering that (a) the ratio between PDD and DLB patients in the dataset was about 3:1, and (b) that the similarity of the cognitive profiles of PDD and DLB patients has been reported to be striking [4], we suspected that the classifiers learned to predict the majority class, rather than effectively discriminating between the two classes.

To check our hypothesis, we built *balanced* datasets, containing the same number of PDD and DLB patients. The first-visit dataset contained 178+178 patients, and the third-visit dataset 155+155 patients. The results are reported in Table 1.

The accuracy ranged between 54% and 58%; it was just slightly superior to the 50% of a random guess. Also in this case, no significant differences were found using either the first-visit or the third-visit data. However, the most important finding of our analysis is that it is not possible, *regardless the used data*, to reliably discriminate between the two diseases starting from the cognitive profiles. The cognitive profiles of the two diseases were so similar that 9 out of the 11 features of the cognitive profile were discretized into a unique bin, i.e. they were useless to discriminate between PDD and DLB.

We think these results to be mainly due to the largely overlapping features of the two diseases; indeed, a recent clinical paper [8] states in its conclusions: "*from the pathologist's point of view, the brains of PDD and DLB patients do not present reliably distinctive features. Therefore, it is probable that in the near future PDD and DLB will be recognized as the same disease with two different courses*".

On the base of these findings, it seems hence advisable to merge these two diseases into a unique class in future classification works.

5.3 Classification Task 2b: {(VAD), (AD)}

By running the classifier on datasets containing all the instances of VAD and AD patients for first and third visit, we measured an accuracy between 71% and 74%. Also in this case, none of the classifiers showed a significant difference in accuracy working on first-visit or third-visit data.

Table 4. Accuracy of different classification algorithms in discriminating between VAD and AD on a *balanced* dataset

	1st visit		**3d visit**		*Significant*
	average	*std. dev.*	*average*	*std. dev.*	*difference?*
classifier					
NAVE BAYES	68.82	5.48	68.38	5.38	NO
BAYES NETWORK	68.27	5.44	68.34	5.37	NO
J4.8 TREE	70.08	4.92	68.96	5.38	NO
SMO	70.14	5.05	69.45	5.16	NO
LOGISTIC REGRESSION	70.18	4.98	69.19	5.11	NO
LAZY.LBR	69.14	5.21	68.77	5.26	NO
CLASS. VIA REGR. (M5)	69.70	4.87	68.58	5.10	NO

The difference between the accuracy recorded on first-visit and third-visit was lower than 2 points of accuracy for each classifier; nevertheless, 6 classifier out of 7 showed a better accuracy on the data of the first visit (2 times such improvement was found to be significant). Overall, all classifiers support hence H_0.

However also in this case, we wanted to avoid the prevalence of VAD patients in the dataset (VAD/AD about 1.6 in the dataset), to bias the experimental results; therefore, we cross-checked these results by running the classifiers also on balanced datasets.

The balanced dataset of the first visit contains 355+355 patients, and that of the third visit 346+346 patients. The accuracy is in this case around 70%, with narrow differences between the different algorithms, as reported in Table 4. Even the differences between the accuracies measured on the first visit data and the third visit data are narrow; although 6 classifiers out of 7 show a slightly higher accuracy on the first-visit data, in no case such difference is statistically significant. Indeed, also these results fully support H_0.

We finally report that a few features (mainly related to simple reaction time and choice reaction time) were discretized into a single bin, being in practice useless for classification.

6 Conclusions

The medical literature acknowledges the need for further research to improve clinical definitions of dementia and to determine the utility of various standardised instruments in increasing diagnostic accuracy, which currently average 58% in DLB, 50% in in VAD and 81% in AD. The ability to apply a single diagnostic assessment to a range of dementias, with good average sensitivity equivalent to, or above that seen with current assessments, is hence a useful addition to existing assessment tools and diagnostic criteria.

This paper provides some clear conclusions regarding the analyis of cognitive profiles for dementia screening via ML techniques; in particular, we (a) looked for robust results, inter-comparing the findings obtained by using different classification algorithms and (b) we integrated our empirical results with the domain-specific literature. Both such approaches are recommended when a previously unexplored domain has to be investigated via ML techniques.

Our experimental results show that with accuracy up to 80% it is possible to discriminate between Parkinson's disease and the two dementia macro-classes (PDD-DLB and AD-VAD). To reliably discriminate between PDD and DLB starting from cognitive profiles is however not achievable; indeed, the actual existence of a significant difference between the two diseases is currently strongly debated within the medical literature. On the basis of these findings, we advice to merge these two classes into a unique class in future classification works. A further classification task of interest is to discriminate between AD and VAD dementia; we show that in this case an accuracy up to 70% can be reached. We have moreover found that a number of variables of the cognitive profile is not useful in discriminating between AD and VAD; hence, machine learning

algorithms appear to be useful also because they show the variables which are sensitive for classification.

A interesting finding that using the first-visit data instead of third-visit data does not lead to any worsening of the classification accuracy. Thus, we can indeed strongly support the use of first-visit data, thus allowing for time and money savings, from both the patients and the company viewpoint.

Acknowledgments

Marco Zaffalon gratefully acknowledges partial support by the Swiss NSF grant 200020-109295/1.

CDR acknowledges the Department of Trade and Industry for partially funding this research through a Global Watch International Secondment grant.

References

1. Simpson, P., Surmon, D., Wesnes, K., Wilcock, G.: The cognitive drug research computerised assessment system for demented patients: A validation study. Int. J. of Geriatric Psychiatry **6** (1991) 95–102
2. Knopman, D.S., DeKosky, S.T., Cummings, J.L., Chui H., Corey-Bloom, J., Relkin, G.: Practice parameter: Diagnosis of dementia (an evidence-based review): Report of the quality standards subcommittee of the American academy of neurology. Neurology **56** (2001) 1143–1153
3. Zaffalon, M., Wesnes, K., Petrini, O.: Reliable diagnoses of dementia by the naive credal classifier inferred from incomplete cognitive data. Artificial Intelligence in Medicine **29**(1-2) (2003) 61–79
4. Ballard, C., Aarsland, D., McKeith, I., O'Brien, J., Gray, A., Cormack, F., Burn, D., Cassidy, T.and Starfeldt, R., Larsen, J., Brown, R., Tove, M.: Fluctuations in attention - PD dementia vs DLB with parkinsonism. Neurology **59** (2002) 1714–1720
5. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers Inc,US (2005)
6. Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning **8** (1992) 87–102
7. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: International Conference on Machine Learning. (1995) 194–202
8. Sellal, F.: Parkinson's disease with dementia and dementia with Lewy body disease: two syndromes, the same disease? Psychogeriatrics **6** (2006) 30–34

When Efficient Model Averaging Out-Performs Boosting and Bagging

Ian Davidson[1] and Wei Fan[2]

[1] State University of New York, Albany, NY 12222
davidson@cs.albany.edu.
[2] IBM T.J. Watson, 9 Skyline Drive, NY 10532
weifan@us.ibm.com

Abstract. The Bayes optimal classifier (BOC) is an ensemble technique used extensively in the statistics literature. However, compared to other ensemble techniques such as bagging and boosting, BOC is less known and rarely used in data mining. This is partly due to BOC being perceived as being inefficient and because bagging and boosting consistently outperforms a single model, which raises the question: "Do we even need BOC in datamining?". We show that the answer to this question is "yes" by illustrating several recent *efficient* model averaging approximations to BOC *can* significantly outperform bagging and boosting in realistic situations such as extensive class label noise, sample selection bias and many-class problems. That model averaging techniques outperform bagging and boosting in these situations has not been published in the machine learning, mining or statistical communities to our knowledge.

1 Introduction and Motivation

The typical aim of classification data mining is to build the most accurate model. Research activity in the machine learning and data mining fields has produced a large number of ensemble techniques such as boosting and bagging that can combine multiple base models to increase accuracy beyond the single best model. However, the use of the classic Bayesian statistical approach to multiple models, Bayesian model averaging, is rarely reported in data mining and even when it is, yields indifferent results [1,5]. The Bayesian approach of using multiple models is to weight each model's belief in a prediction ($P(y_i|\mathbf{x},\theta)$) by the model's posterior probability ($P(\theta|D)$) and is known as Bayesian model averaging in the statistical literature [11] and the Bayes optimal classifier (BOC) in the machine learning and data mining literature [14]. The techniques we shall use in this paper are approximations to the BOC since we do not average over all models. Given a test set instance, $\mathbf{x}$, to be classified into one of k classes ($y_1 ... y_k$), a training dataset D and model space Θ the BOC chooses the class that maximizes equation (1).

$$argmax_{y_i} : P(y_i) = \int_{\theta \in \Theta} P(y_i|\mathbf{x},\theta) P(\theta|D) d\theta \quad (1)$$

The BOC is claimed to be optimal for a number of reasons under specific conditions: Kohavi claims it reduces bias and variance [12] while Buntine claims

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 478–486, 2006.
© Springer-Verlag Berlin Heidelberg 2006

it prevents overfitting [1] and Domingos [5] states "no other method can consistently achieve lower error rates". In addition there is significant experimental evidence supporting its effectiveness in the statistical literature [11]. Yet there seems to be little interest in using model averaging in data mining. Instead the two most common ensemble model techniques are bagging and boosting [3]. These techniques are rightfully extensively used as they are relatively easy to implement, have been shown to work for a variety of learners and a variety of data sets. In contrast model averaging is typically more difficult to implement, particularly since BOC in its rudimentary form requires performing an integration over the entire model space which is computationally prohibitive for the *complex* model spaces such as trees and large datasets used in the data mining. Compared to model averaging, ensemble techniques such as bagging and boosting **combine** not average and the base unit is a **vote** not a probability. Minka [15] succinctly notes the difference that BOC is a method of "soft model selection" using multiple models when there is *uncertainty* to which model is the best model. For example, if all posterior probabilities are the same then model uncertainty is maximum. In contrast techniques such as bagging and boosting are methods to *combine* multiple models to create a new (and potentially more appropriate) model space than the base model space [15].

2 When Averaging Will Outperform Combining

In this section we compare and contrast model averaging and model combination and conclude the section by describing data set conditions where model averaging should perform better than model combination. In later sections we empirically test these claims. It may appear that model averaging and techniques such as boosting and bagging are similar but how the multiple models are used are quite different. Many ensemble techniques explicitly or implicitly combine multiple models. For example, the serial nature of boosting to focus on misclassified instances *effectively* means that the original tree built from the unweighted training set has subsequent additional trees grafted onto the leaf nodes. In this way the model space boosting is searching is a combination of the base model class (trees). Similarly, it has been noted that bagging models can create a more express model space than the base model space [15]. In contrast model averaging never explicitly combines models. Rather, each model's predictions are weighted by the belief (posterior) that it is the true model. Furthermore, in the presence of excessive number of training instances equation 1 will simply have most of the posterior mass on a single model and perform no better than a single tree. As the statistical literature [11] and Minka [15] note model averaging should perform well when there is substantial model uncertainty where best-model uncertainty can be quantified as being one minus the posterior probability of the most probable model. We note that there are other general measures of model uncertainty such as the entropy in the posterior [2]. Formally:

Definition 1. ***Best-Model Uncertainty*** *is the degree of belief that the most probable model (θ^*) is* ***not*** *the best model in the model class (Θ). That is:* $ModelUncertainty(\Theta, D) = argmin_{\theta \in \Theta}\ (1 - P(\theta|D))$

When model uncertainty exists, it is because there is insufficient data to conclusively state that one model is the best and hence building combinations of models can be perceived as building an overly complex model given the amount of data. We see from definition 1 that model uncertainty is likely to exist if there is no highly probable model. By a simple expansion of $P(\theta|D) = \frac{P(\theta)P(D|\theta)}{P(D)}$ using Bayes theorem we see that this can occur if the numerator, particularly the likelihood is small. In decision tree learning this can occur for a number of reasons. Since each tree path forms a distinct part of a model we can say that $P(D|\theta) = \Pi_{1\ldots n} P(D_i|Leaf(D_i))$ where $Leaf(D_i)$ is a function returning the leaf the i^{th} instance is assigned to. The term $P(D_i|Leaf(D_i)) = \frac{Count(Class(D_i), Leaf(D_i))}{Number(Leaf(D_i))}$ where $Count(Class(D_i), Leaf(D_i))$ returns the number of training set instances at the leaf having the same label as D_i and $Number(Leaf(D_i))$ the total number of instances at the leaf node. The leaf nodes of the tree may be poorly or incorrectly populated for any number of reasons. The (non-exhaustive) set of data set conditions we believe where this will occur and shall explore in this paper are: 1) Training sets with excessive class label errors (i.e. security/fraud applications where labeling is difficult), 2) High dimensional data but a relatively few number of instances (i.e. bioinformatics problems) and 3) Multi-class problems involving a large number of classes (i.e. fault diagnosis).

In addition, even if model uncertainty is not particularly great, it may be worth removing by averaging. In this paper we shall explore a situation discussed in our recent work, sample selection bias [7]. Since the training and test data sets are drawn from different distributions, even a highly probable model for the training set may produce poor predictions on the test set.

3 Efficient Model Averaging Techniques

Random Decision Trees (RDT). RDT were originally proposed in [8]. Rather than using purity functions (i.e. information gain) like traditional decision tree algorithms, RDT chooses a feature/column to split on *randomly.* A discrete feature is chosen only once in each decision path. A continuous feature can be chosen multiple times in the same decision path with a different decision threshold each time. Since both feature and decision thresholds for continuous features are chosen randomly, each RDT is likely to be different. The tree stops growing if either the number of instances at the current node is zero or the depth of the tree exceeds some predefined limit. In our experiments, the depth of the tree is limited to be up to the number of features in order to give each feature equal opportunity to be chosen in any decision path. During classification, each random tree computes a posterior probability at the leaf node. That is, if there are n examples at a leaf node and q_i belong to class label y_i, the posterior probability $P(y_i|\mathbf{x}, \theta) = \frac{q_i}{n}$. For a given test instance, the posterior probability

outputs from multiple decision trees are **averaged** to produce a final probability estimate with the most probable class for an instance being the prediction. As found in [8], typically 30 random trees give satisfactory results and as little as 10 random trees produce results better than using a single traditionally grown decision tree [8]. RDT have been shown to have accuracy comparable to or higher than bagging and random forest but at a fraction of the training cost for a variety of data sets including those with many irrelevant attributes [8,9]. Though with RDT each *tree structure* is created randomly the leaf probabilities are estimated from the training data set. Therefore, in the limit as the number of trees approaches infinity, RDT effectively computes the following: $P(y_i|\mathbf{x}, \Theta) = \sum_{\theta \in \Theta} P(\theta|D).\frac{q_{i,\theta}}{n_\theta}$ RDT though superficially similar to random forest of trees (RFT) are different in a number of ways (see [9] for details). Furthermore, RDT are different to the random trees referred to by Dietterich [3] as in that work the splits are randomly chosen from the twenty most informative splits.

Parametric Bootstrap Model Averaging. The bootstrap model averaging approach [2] is a frequentist approach to model averaging. The philosophy of model averaging is to remove model uncertainty that arises due to **a single finite data set**. The typical view of mining is that single training set of size n is available from the underlying distribution that generated the data F. This masks the underlying uncertainty that the training data is one of many that could have been chosen/generated. Building a model for each possible data set would create a distribution which is the frequentist analog to the posterior distribution but does not use a prior. However, typically we do not have the luxury of many different data sets drawn independently of the same size so we can not compute the uncertainty over the model space from them. To approximate this distribution using a *single data set*, Efron [6] created the non-parametric and parametric bootstrapping approaches. Non-parametric bootstrapping which has been used extensively by the machine learning community is an example of attempting to simulate draws from F when no knowledge of its form is known. Parametric bootstrapping which has received little attention is used when some underlying knowledge on the form of F is known. In this paper we make use of a simple generative parametric model of F which assumes that the independent variables are conditionally independent given the dependent variable value. The parameters for this model are estimated from the data and virtual training examples are drawn/bootstrapped from it to build models. More complex models of F could be used and remains an active research area. Formally the bootstrap model averaging approach is: $P(y_i|\mathbf{x}, \Theta) = \sum_{D', \theta_i = L(D')} P(y_i|\mathbf{x}, \theta_i) P(D'|D)$ In this paper, the learner, L, used with PBMA is J48 (Weka's equivalent to C4.5) to produce trees built with the default parameters.

4 Experiments

We now illustrate the performance of model averaging techniques and other ensemble techniques when considering model uncertainty is beneficial (see section

2) and find that bagging and boosting do not perform well. We use the Weka software to perform bagging and AdaBoost using the J48 decision tree base learner (equivalent to C4.5). This requires changing Weka slightly as the default implementation of bagging in Weka aggregates conditional probabilities and *not* votes [10]. In all experiments, regardless of the ensemble technique (PBMA, Boosting, Bagging), the default parameters were used for J48. We randomly divide the data into 80% for training and the remaining 20% for testing unless otherwise stated.

Model Uncertainty Due to Class Label Noise

If there exists a deterministic relationship between the independent variables and class label, and the model space is suitable, it is likely any reasonable learner will find a highly probable model that can correctly map feature vectors of independent variables to their correct class labels. Hence the model uncertainty would be relatively small. However, the addition of class label noise, i.e., the correct class label being flipped to another class, would almost certainly reduce the probability of the "correct model" otherwise trained from dataset without label noise, hence increasing model uncertainty. We took several UCI data sets and introduced 20% class label noise in the training sets only by randomly flipping 20% of class labels. We then try the model averaging approaches and compare them against the other ensemble techniques. To illustrate the efficiency of PBMA and RDT we use only 30 models but use 100 models for bagging and boosting. Table 1 (left) shows that the model averaging techniques outperform the other ensemble techniques and single models in each situation. A pair-wise student t-test using the same training and test divisions, shows that the poorer performing model average technique, PBMA, outperforms both bagging and boosting at the 99% confidence interval for all data sets. As expected boosting performs quite poorly as it apparently continues to build models to fit the added noise while bagging can only occasionally significantly outperform a single model.

Biased Training Sets

Recent work by Zadrozny and ourselves [17,7] has classified many learners including decision trees and naive Bayes as global learners that can be affected by sample bias. In particular the sample bias investigated is that the probability of the instance being selected in the training data (denoted by the event, $s = 1$) is conditionally independent of the instance label (y) given the instance's description (x), formally: $P(s = 1|x, y) = P(s = 1|x)$. This type of sample bias is effectively when instances are not chosen at random to be in the training set but instead depending on their description but not their label. The test set is drawn randomly. This sample bias occurs prevalently in applications where the occurrence of particular instances' change but not the concept (i.e. relationship between x and y). For the rest of this paper when we refer to **sample bias**, we refer to this type of bias. Decision trees are known to be unstable, and is categorized as "global" classifier in [7]. The structure of decision tree

is sensitive to sample selection bias, which makes it unlikely to find that most probable decision tree otherwise trained from dataset without sample selection bias.

Artificial Sample Selection Bias. We can artificially introduce sample bias into the previously mentioned UCI data sets by first dividing the data into training and test sets. We then sort only the training set on the attributes and remove the first 10% of all instances. Note this introduces a training sample bias but does not change the relationship between the independent variables and class-labels as our previous experiments did. Table 1 (right) shows that model averaging performs better than other ensemble techniques even though only 30 models are used and 100 models are used for bagging and boosting. Unlike the previous situation, boosting performs better than bagging on average **but both perform worse than a single tree**.

Sample Selection Bias in Newsgroup Classification. We now focus on the newsgroup data where the training and test data sets are drawn from similar but not identical distributions. We perform experiments on the 20 Newsgroups [16] datasets using the standard bydate version division into training (60%) and test (40%) based on posting date. The division creates a temporal bias. For example, in the GUNS newsgroup the word "Waco" occurs extensively in news articles in the training set but not in the test set as interest in the topic fades. However, since the proportion of each class label is the **same** in the training and test data sets there is no class label bias. We used the tool Rainbow [13] to extract features from these news articles. The feature vector for a document consists of the frequencies of top ten words by selecting words with highest average mutual information with the class variable. To better understand the performance of model averaging we treat the problem as binary classification between a variety of newsgroups. Table 2 illustrates the improvement that model averaging provides. Again, the better performance is obtained using only 30 models for the model averaging approaches as opposed to 100 models for bagging and boosting. In this situation boosting performs better than bagging but the model averaging approaches outperform both and the single best model.

Table 1. Average error of various techniques on standard UCI data sets. Left: With class label noise in training set only over one hundred repetitions. Right: With biased training sets over one hundred repetitions. UP=Unpruned, P=Pruned.

	Breast	**Vote**	**Pima**	**Credit**	**Wine**	**Breast**	**Vote**	**Pima**	**Credit**	**Wine**
UP. Tree	10.6	5.2	31.8	36.5	37.6	1.5	3.7	26.5	3.1	35.3
P. Tree	4.5	12.6	31.8	37.4	37.0	2.1	3.7	27.6	12.2	35.3
RDT	**0.5**	3.7	**30.0**	**30.6**	**26.4**	**0.5**	**2.7**	26.1	**1.0**	**27.8**
PBMA	2.5	**2.2**	30.9	30.7	26.7	1.0	3.1	**25.7**	1.3	28.0
Bagging	4.0	4.2	33.9	34.3	34.3	4.04	3.7	28.0	5.61	38.2
Boosting	15.7	19.3	34.0	42.4	38.2	2.5	4.4	28.4	5.02	38.1

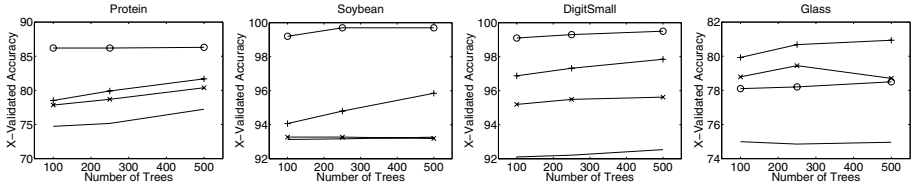


Fig. 1. Accuracy versus No. trees: Bagging(-), Boosting(-x-), RDT(-o-) & PBMA(-+-)

Table 2. Error of various techniques on newsgroup data. Training set is first 60% of the year's postings and the test set the last 40%.

	BaseBall Hockey	**Christian Sale**	**M.Cycle Guns**	**MidEast Guns**	**MidEast Electrical**	**Mac. Religion**
Unpruned Tree	15.7	7.9	10.5	20.3	14.4	18.4
Pruned Tree	15.7	7.4	10.2	20.3	14.4	18.7
RDT	**11.9**	6.6	**8.5**	10.5	**7.2**	12.7
PBMA	12.0	**6.0**	8.6	**9.8**	7.4	**11.9**
Bagging	14.8	7.9	10.5	20.3	14.4	18.4
Boosting	12.6	7.7	9.2	10.7	11.7	13.2

Model Uncertainty Due to Large Numbers of Classes
We examine four data sets where the number of classes is large: Digit (10), Glass (6), Protein (6), Soybean (19). Furthermore, the number of columns is relatively high and instances relatively small. For each data set the number of columns and instances is: Digit (17, 469), Glass (10, 214), Protein (21, 116) and Soybean (36, 683) respectively. Assuming a uniform distribution of classes amongst the instances there are as little as 19 instances per class (not leaf) and hence model uncertainty is likely to be high. We calculated the ten-fold cross-validated accuracy ten times for bagging, boosting, RDT and PBMA on these four data sets building, 100, 250 and 500 trees. We found that the poorer performing model average technique with respect to performance over all data sets, PBMA, outperforms both bagging and boosting for all data sets at the 99% confidence interval. When comparing the model combination approaches bagging, boosting to the model averaging approaches RDT and PBMA we find that both averaging approaches work significantly better except for the Glass data set when only PBMA outperform both combination techniques (see Figure 1).

5 Conclusion and Future Work

Model averaging is a method of choice for statisticians to make use of multiple models. However, model averaging is rarely used in data mining and instead other ensemble techniques such as boosting and bagging are used extensively. We believe this is for two primary reasons: a) Model averaging is perceived as

being computationally inefficient and b) boosting and bagging have been shown to work for a variety of problems. A valid question is then: "Does the data mining practioner require model averaging as an ensemble technique?". The answer to this question is yes, there are situations where model averaging can significantly outperform both bagging and boosting using two recent efficient approaches to model averaging. This is because a) boosting and bagging are model *combination* not averaging approaches and b) model averaging works best when there is model uncertainty. The situations where averaging to remove model uncertainty is beneficial we explored in this paper include when the number of classes is large, excessive class label noise and training sample bias occurs. We believe that these are not academic situations but will occur in fields such as bioinformatics and security and demonstrated that one of the situations (training sample bias) occurs in the newsgroup data sets. In this work we identified situations where model averaging performs well, but we empirically saw that boosting and bagging performed badly. For some of these situations (i.e. class noise) and techniques (boosting) it is well known why the technique performs badly, it would be interesting to explore why bagging and boosting failed in these other situations.

Acknowledgements. We thank the reviewers for their helpful comments.

References

1. Buntine W. (1990). *A Theory of Learning Classification Rules*, Ph.D. Thesis, UTS.
2. Davidson I. (2004). An Ensemble Technique for Stable Learners with Performance Bounds, *AAAI 2004*.
3. Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40 (2).
4. Domingos, P., (1997). Why Does Bagging Work? A Bayesian Account and its Implications. *KDD 1997*.
5. Domingos P. (2000). Bayesian Averaging of Classifiers and the Overfitting Problem, *AAAI 2000*
6. Efron, B. (1982). The jackknife, the bootstrap, and other resampling plans. SIAM Monograph 38.
7. Fan W., Davidson I., Zadrozny B. and Yu P., (2005) An Improved Categorization of Classifier's Sensitivity on Sample Selection Bias, *ICDM 2005*.
8. Fan W., Wang H., Yu P.S, Ma S., (2003). Is random model better? On its accuracy and Efficiency, *ICDM 2003*.
9. Fei Tony Liu, Kai Ming Ting, Wei Fan, (2005). Maximizing Tree Diversity by Building Complete-Random Decision Trees. *PAKDD 2005*.
10. Frank, Eibe, *Personal Communication*, 2004.
11. Hoeting J., Madigan D., Raftery A., and Volinsky C., (1999). "Bayesian Model Averaging: A Tutorial", *Statistical Science 14*.
12. Kohavi R., Wolpert D., (1996). Bias Plus Variance Decomposition for 0-1 Loss Functions, *ICML 1996*.
13. McCallum, A. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/~mccallum/bow.

14. Mitchell T., (1997). *Machine Learning*, McGraw-Hill.
15. Minka, T.P., Bayesian model averaging is not model combination, MIT Media Lab note (7/6/00), http://research.microsoft.com/~minka/papers/bma.html
16. Rennie, J. (2003) *20 Newsgroups*. Technical Report, Dept C.S., MIT.
17. Zadrozny B., (2004). Learning and evaluating classifiers under sample selection bias, ICML 2004.

Peak-Jumping Frequent Itemset Mining Algorithms

Nele Dexters[1], Paul W. Purdom[2], and Dirk Van Gucht[2]

[1] Departement Wiskunde-Informatica, Universiteit Antwerpen, Belgium
nele.dexters@ua.ac.be,
[2] Computer Science Department, Indiana University, USA
pwp@cs.indiana.edu, vgucht@cs.indiana.edu

Abstract. We analyze algorithms that, under the right circumstances, permit efficient mining for frequent itemsets in data with tall peaks (large frequent itemsets). We develop a family of level-by-level peak-jumping algorithms, and study them using a simple probability model. The analysis clarifies why the jumping idea sometimes works well, and which properties the data needs to have for this to be the case. The link with Max-Miner arises in a natural way and the analysis makes clear the role and importance of each major idea used in this algorithm.

1 Introduction

The frequent itemset mining (FIM) problem [1,2] is to determine combinations of items that occur together frequently: frequent itemsets (FIS). In a store with n items visited by s shoppers, the question is which of the 2^n possible itemsets are bought together by at least k shoppers, with k the threshold.

The *support* of itemset I, $supp(I)$, is the number of shoppers that buy all items from I. Itemset I is *frequent* if $supp(I) \geq k$, and *infrequent* otherwise. If $J \subseteq I$, then $supp(J) \geq supp(I)$. This is the *monotonicity property* of the support function. If I contains m items and (1) if I is frequent, then all of its 2^m subsets are also frequent, and, (2) if I is infrequent, then all of its 2^{n-m} supersets are also infrequent. These principles are at the core of FIM algorithms because of their potential to reduce the search space. Max-Miner [3] is the canonical example of an algorithm that uses pruning principle (1) and Apriori [2] of principle (2).

Datasets with large FIS are called *peaky*. Algorithms designed to run on peaky data, such as Max-Miner, MAFIA [4], certain versions of Eclat [10] and FP-Growth [8], and GenMax [7] usually avoid outputting every FIS. Instead, they can concentrate only on the maximal FIS. The basic idea for dealing efficiently with large FIS is trying to jump up the peaks. Early in the processing, it is important to determine the frequency of selected large itemsets. When a large FIS is found, the challenge is to quickly avoid processing its subsets.

This paper uses simple probabilistic analysis to show that if you want (1) to do FIM, (2) to handle peaky data efficiently, and (3) to use only simple ideas, you are more or less forced into an algorithm that is very close to Max-Miner [3]. An alternate view is that if any key idea is omitted from Max-Miner, then its performance will be much poorer for some simple real-world datasets.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 487–494, 2006.
© Springer-Verlag Berlin Heidelberg 2006

2 Two Shopper, Two Item Types Data Model

The two shopper, two item types data model is based on the following notation:

- s_i shoppers of type i $(i = 1, 2)$,
- n_j items of type j $(j = 1, 2)$, and
- probability p_{ij} that a shopper of type i buys an item of type j.

By convention, p_{11} is the largest of the probabilities.

This shopping model brings out the most important aspects of peak-jumping algorithm performance. The probability that a shopper of type i $(i = 1, 2)$ buys m_1 different items of type 1 and m_2 of type 2, regardless of what else is bought:

$$P_i(m_1, m_2) = p_{i1}^{m_1} p_{i2}^{m_2}. \tag{1}$$

The expected support of an itemset that contains m_1 particular items of the first kind and m_2 of the second:

$$s_1 P_1(m_1, m_2) + s_2 P_2(m_1, m_2), \tag{2}$$

$$s_1 p_{11}^{m_1} p_{12}^{m_2} + s_2 p_{21}^{m_1} p_{22}^{m_2}. \tag{3}$$

3 The Perfect Jump Algorithm

3.1 Perfect Jumps

Suppose an algorithm jumps and finds that some itemsets on a high level are frequent. The question is how to make good use of this information. In principle, monotonicity can be used. If we jump from level l to level x $(x > l)$ and find some FIS on level x, then the only candidates that need to be considered on level $l+1$ are formed from pairs of sets on level l that are not subsets of the FIS on level x. Otherwise, the candidate is frequent based on monotonicity. Unfortunately, there is no fast general algorithm to carry out this subsumption test.

This paper focusses on the idea: a jump to level x is successful if and only if *every* candidate on level x is frequent. We call this a *perfect jump*. Now, every candidate between levels $l+1$ and x is frequent. By only allowing perfect jumps, the performance of the algorithm is limited, but the time for processing levels l and x is proportional to the sum of the number of candidates on the two levels.

Suppose that on level l the FIS contain a_l distinct items. If all the FIS are coming from a single peak, the itemset causing the peak contains a_l items. Furthermore, the number of FIS on level l is $\binom{a_l}{l}$. When the number of FIS is not given by this formula, there is no possibility that a perfect jump will occur.

Suppose, in a perfect jump case, we jump to level x. There are $\binom{a_l}{x}$ candidates on level x. They are formed by taking each combination of x items from the a_l items that occur in the FIS on level l. If any early jump is successful, then the jumping algorithm can be much faster than an algorithm that does not try to jump. In particular, the jumping algorithm does not have to explicitly determine the frequency status of the $\sum_{j=l+1}^{x-1} \binom{a_l}{j}$ itemsets that occur strictly between levels l and x. Clearly, the biggest savings come from doing a jump from a level l, small compared to a_l, to a level x near a_l. Therefore, we will tacitly assume that jumps occur from small levels to high levels.

3.2 The Algorithm

Based on the concept of perfect jumps, we present the *Perfect Jump Algorithm*.

1. (*Start*) Set $l = 0$. If the number of shoppers is greater than k then set the empty set to frequent, otherwise set the empty set to infrequent.
2. (*Check if done*) If there are no FIS on level l then stop. Otherwise set $l = l+1$.
3. (*Process level* l) Determine the frequency of the itemsets of size l that can be built from the FIS from level $l-1$. (See [2] for details on how to do this.)
4. (*Try to jump*) Set a_l to the number of items that appear in the FIS on the current level. If the number of FIS on this level is strictly smaller than $\binom{a_l}{l}$ or $l \geq a_l/2$ then go to Step 2.
5. (*Try jumping*) Set $x = a_l - l$. For every itemset that can be formed by choosing x of the a_l items, determine whether or not the item set is frequent. If every such itemset is frequent, then set $l = x + 1$ and go to Step 3.
6. (*Continue*) Go to Step 2.

When the conditions for attempting to jump are met, the Perfect Jump (PJ) Algorithm tries jumping to level $a_l - l$ because the number of candidate itemsets to be tested at that level is the same as the number of FIS on level l. This balances the work attempting jumps with the other work, thereby ensuring that the algorithm is never slower than Apriori by more than a factor of two. In favorable cases, i.e., when a large jump occurs, the PJ Algorithm is much faster.

4 Analysis of the Perfect Jump Algorithm

The shopping model from Section 2 generates a distribution of datasets. Every dataset with $s_1 + s_2$ shoppers and $n_1 + n_2$ items will be generated with some probability, but some of these sets have much higher probability than others. We now consider a particular dataset in which the actual supports of the itemsets equal the expected supports from the probability model.

To have a perfect jump, we must have a tall peak. Since p_{11} is the largest probability, eqs. (1, 2) imply that there is no peak unless p_{11} is close to one.

We begin the analysis of the performance of the PJ Algorithm for the case where the peak at level $x = m_1 - l$ is formed only from m_1 items of type 1.

$$s_1 P_1(m_1 - l, 0) + s_2 P_2(m_1 - l, 0) \geq k, \tag{4}$$

$$s_1 p_{11}^{m_1 - l} + s_2 p_{21}^{m_1 - l} \geq k. \tag{5}$$

To meet the condition for a perfect jump the FIS on level l must be those itemsets that contain l items of type 1 and no items of type 2. Obeying eq. (5) already ensures that every itemset with l items of type 1 is frequent. To ensure that no other itemsets are frequent, we need that for every r $(1 \leq r \leq l)$,

$$s_1 P_1(l - r, r) + s_2 P_2(l - r, r) < k, \tag{6}$$

$$s_1 p_{11}^{l-r} p_{12}^{r} + s_2 p_{21}^{l-r} p_{22}^{r} < k. \tag{7}$$

The first term on the left side of eq. (7) is biggest at $r = 1$, since $p_{11} > p_{12}$. The second term is biggest at $r = 1$ when $p_{21} \geq p_{22}$ and at $r = l$ when $p_{21} \leq p_{22}$.

We now start a case analysis of the condition for a perfect jump to occur on level l. The cases are selected for the most interesting parameter settings.

For the first case, assume that $p_{21} = 0$ and $p_{22} = 0$. To have a perfect jump we need to have

$$p_{11}^{m_1 - l} \geq \frac{k}{s_1} \tag{8}$$

and

$$p_{12} < \frac{k}{s_1 p_{11}^{l-1}}. \tag{9}$$

If we write $p_{11} = 1 - \epsilon$, then logarithm of eq. (8) says

$$(m_1 - l)\,(-\epsilon + \text{higher order terms}) \geq \ln(k/s_1), \tag{10}$$

$$\epsilon \leq \frac{-\ln(k/s_1)}{m_1 - l} \qquad \text{(neglecting higher order term)}. \tag{11}$$

Essentially, to have a perfect jump from a low level, p_{11} must be close to 1 and p_{12} must be less than k/s_1. For a particular dataset eq. (9) says that threshold k has to be set large enough to filter out the noise from the items that are not part of the peak, and eq. (8) determines whether there is any value of k that works. The value of l plays only a small role in what is happening[1]. Thus, if this noise resulting from shoppers buying the peak items and occasional not-peak items causes trouble on level 1, it probably will cause trouble on the higher levels too.

In a second case, we assume that p_{21} and p_{22} are not both 0 but neither is close to 1. In this case, we may not be able to make perfect jumps for small values of l, but since in this case the second term in eq. (7) decreases rapidly with l and the second term in eq. (5) only helps, we will be able to make a perfect jump with a moderate value of l.

For a third case, we assume that p_{21} is large. When this is the case, both types of shoppers contribute to the peak. We need both p_{12} and p_{22} to be small (close to 0) to have a perfect jump to a peak based on items of just type 1.

A next case is considered, when three or four of the probabilities are large. We now may have a peak based on all the items of both types. When s_1 is small, s_2 is large, and p_{22} is large, we may have a peak based just on items of type 2.

In a last case, when p_{11} and p_{22} are near 1 but p_{12} and p_{21} are near 0, the model is generating data with two peaks. The PJ Algorithm needs an additional idea before it can do well on such data (See Section 6).

To summarize this section, the PJ Algorithm will do well on data with a single peak so long as the subset of shoppers that are buying most of the peak items are unlikely to buy any non-peak items. This is the case even in the presence of other shoppers, so long as their buying patterns do not lead to other peaks.

[1] If p_{11} is not extremely close to 1 (eq. (11)), we don't have a peak. If p_{11} is quite close to 1, then p_{11}^{l} [1] is close to 1 for small l.

5 The Perfect Jump Algorithm with Lower Bounds

If we know the frequency of all FIS on levels $l-1$ and l, we can obtain a lower bound on the frequency of the candidates on higher levels using [3]:

$$Supp(I \uplus J) \geq \sum_{j \in J} Supp(I \uplus \{j\}) - (|J|-1)Supp(I), \tag{12}$$

in which $\uplus$ denotes a union on disjoint sets. For us, I is a set on level $l-1$ and J is the set of items other than those in I that make up the peak.

We can modify the PJ Algorithm so that Step 5 replaces the actual count on level $x = m_1 - l$ with the bound given by eq. (12) (where $|I| = l-1$ and $|J| = m_1 - l - |I|$). To analyze the conditions for a successful perfect jump using this modified algorithm with data with a peak based only on items of type 1, we still need eq. (9), but we need to replace eqs. (4, 5) with

$$\begin{aligned} &s_1[(m_1-2l+1)P_1(l,0) - (m_1-2l)P_1(l-1,0)] \\ &\qquad + s_2[(m_1-2l+1)P_2(l,0) - (m_1-2l)P_2(l-1,0)] \geq k, \qquad (13) \\ &s_1 p_{11}^{l-1}[(m_1-2l+1)p_{11} - (m_1-2l)] \\ &\qquad + s_2 p_{21}^{l-1}[(m_1-2l+1)p_{21} - (m_1-2l)] \geq k. \qquad (14) \end{aligned}$$

For the case where $p_{21} = 0$, if we write $p_{11} = 1-\epsilon$, and carry out the calculations to first order in ϵ, we obtain

$$(1-\epsilon)^{l-1}[1-(m_1-2l+1)\epsilon] \geq \frac{k}{s_1}, \text{ leading to } \epsilon \leq \frac{1-k/s_1}{m_1-l}. \tag{15}$$

Suppose we write $k/s_1 = 1-\delta$. For datasets associated with the probability model, when k is almost as large as s_1 (δ near 0), the PJ Algorithm needs p_{11} equally close to 1 whether it uses estimates or exact counts in order for the jump to be successful. When k is less than s_1 by a few powers of ten, the version of the algorithm that uses exact counts will succeed in making perfect jumps with values of p_{11} that are much further from 1.

6 The Perfect Jump Algorithm with Connected Components

Items with intermediate probability interfere with the performance of the PJ Algorithm. When this interference is coming from a second type of shoppers, it can often be eliminated by breaking the data into disjoint components.

Define FIS I and J as *connected* when

$$|I| = |J| = |I \cap J| + 1. \tag{16}$$

The resulting connectedness relation is symmetric, but not transitive[2]. The reflexive transitive closure is an equivalence relation. Therefore, we can divide the

[2] This relation was previously used in some versions of the Eclat Algorithm [10].

FIS on level l into equivalence classes by putting into the same class any two frequent itemsets that are connected by a chain. Itemsets from two different equivalence classes never interact in FIM algorithms. Indeed, if I and J are FIS at level l and in different equivalence classes, then itemset $I \cup J$ can not be frequent, because if it were, there would be a chain connecting I and J.

Thus, the PJ Algorithm remains correct if we partition the FIS on level l into equivalence classes and process each equivalence class independently. On level 1, all FIS are in the same class, but on higher levels there can be many classes.

For our random model, it is interesting to consider the conditions where the itemsets made up of items of type 1 are in a different equivalence class than those of type 2. To have two classes on level l, all those sets made up of $l-t$ items of type 1 and t items of type 2 with $1 \leq t \leq l-1$ must be infrequent. The expected number of such sets is

$$s_1 P_1(l-t,t) + s_2 P_2(l-t,t). \tag{17}$$

When this number is significantly below k the typical dataset from the model will have two equivalence classes. Thus the condition for each peak to be in its own equivalence class is the same eq. (7) except that eq. (7) permits $t = l$ while eq. (17) does not allow that case. The $t = l$ case is associated with a peak that will be in a separate component.

We see that by combining the idea of perfect jumps with partitions, we obtain a simple algorithm that is efficient at finding the maximal FIS in some interesting datasets with several large peaks.

7 The Max-Miner Algorithm

The PJ Algorithm with Connected Components is a fine algorithm except for datasets with several items of intermediate probability. One way to overcome this problem is to partition the search space of itemsets so that many of the partitions don't have itemsets with items of intermediate probability. This is the first key idea of the Max-Miner Algorithm [3].

Max-Miner is a level-wise FIM algorithm that works like Apriori, except that it also has peak-jumping capabilities. Throughout its running, it maintains a set $\mathcal{M}$ which at the end will contain all the maximal FIS.

At the start, Max-Miner picks a total ordering σ on the set of all items. It then begins its search for FIS. Assume that the algorithm has discovered the set $\mathcal{F}_{l-1}$ of all FIS at level $l-1$. From these sets, it generates the set $\mathcal{C}_l$ of all candidates at level l. Based on σ and $\mathcal{F}_{l-1}$, it then partitions $\mathcal{C}_l$ as follows. The ordering σ induces a lexicographic order on the itemsets in $\mathcal{F}_{l-1}$, say $I_1 < I_2 < I_3 < \cdots$. For I_1, the algorithm finds all candidates that contain I_1 and puts them in a cell, called $cell(I_1)$. Then, from the *remaining* candidates, it finds all those that contain I_2 and puts them in $cell(I_2)$. It then determines $cell(I_3)$ etc. Notice that some of these cells may be empty. After this partitioning phase, the algorithm processes each non-empty cell as follows. For each itemset in cell $cell(I)$, the algorithm determines whether or not it is frequent. If it finds FIS, it constructs

itemset J_I which consists of all the items that occur in $cell(I)$, except those that are in I. Call $peak(I)$ the set $I \cup J_I$. The algorithm then enters its peak-jumping phase. For each $peak(I)$ it determines whether or not it is frequent (either by explicit counting or by using the lower bound formula). If it is frequent, $peak(I)$ is added to $\mathcal{M}$, otherwise the FIS in $cell(I)$ are added to $\mathcal{M}$. Then the algorithm moves from level l to level $l + 1$.

When comparing Max-Miner with the PJ Algorithm, think of Max-Miner as a parameterized version of the PJ Algorithm (relative to I) which attempts only jumps from level 1 to the highest possible level (as determined by J_I).

In practice, Max-Miner uses a more subtle strategy to update $\mathcal{M}$. When it adds $peak(I)$, it removes from $\mathcal{M}$ all the itemsets that are subsumed by it. And, when it considers adding the FIS in $cell(I)$, it only adds those that are not subsumed by an existing itemset in $\mathcal{M}$. Clearly, under this update strategy, there is the guarantee that at the end, $\mathcal{M}$ consist exactly of all maximal FIS. Clearly this can be costly because each of the subsumption tests has to be done explicitly, and no general algorithm exists that can do this efficiently. Regardless of which update strategy is used, in the cases where $peak(I)$ is frequent and J_I is large, Max-Miner can be expected to gain substantial performance improvements (see the experimental results in [3]).

If the initial ordering σ picked by Max-Miner is a randomly selected ordering, then the FIS at level $l - 1$, i.e. those in $\mathcal{F}_{l-1}$, will also occur in some random order. When the algorithm then enters its partitioning phase of the candidates at level l, i.e. $\mathcal{C}_l$, it can be expected that for I in $\mathcal{F}_{l-1}$, the set J_I will have a mixture of low- to high-probability items. Even if J_I is large, the analysis in Section 4 strongly suggests that $peak(I) = I \cup J_I$ will be infrequent and the algorithm will not be able to gain much from its peak-jumping abilities. Overcoming this problem is the second key idea of Max-Miner. Rather than picking a random ordering, it picks an ordering based on the supports of the items. Specifically, the ordering is determined by arranging the items in *increasing* order of their supports. This ordering is much more likely to decrease the generation of many J_I's which have a mixture of low- to high probability items. Consequently, the algorithm considerably improves its opportunity to obtain significant performance improvement on account of peak-jumping.

Max-Miner searches for FIS in level-wise (breadth-first) order. If one changes to depth-first order, it is necessary to give up the full Apriori candidacy test. Doing so leads to Eclat-like versions of Max-Miner [4,8,10,7]. However, with respect to the performance gain due to peak-jumping, these Eclat-like algorithms get essentially the same benefits as Max-Miner.

8 Discussion and Conclusion

When considering algorithms to process peaky data, it is necessary to consider the variation of probability of various items and the correlations in the data.

If the data has some low frequency items, some high frequency items, no intermediate frequency items, and no important correlations, then the data may have one peak. The P J Algorithm has good performance in this case.

If the data is the same as above but with important correlations, then there may be several peaks. The PJ Algorithm needs to be augmented with a division into independent components before it can handle such data efficiently.

These ideas still do not lead to an efficient algorithm if the dataset has several items with intermediate probability. Many datasets have the property that the occurrence of items is close enough to independent that the relative probability of items on one level is similar to that on other levels. In such cases, one can do jumps relative to an itemset that is missing the items of intermediate probability. This is the essential reason why Max-Miner types of algorithms are efficient.

This line of reasoning makes it clear that ordering items from least-frequent to most-frequent is critical to the success of Max-Miner. If for example, Max-Miner had processed items in random order, it is unlikely that it would produce an itemset that had all of the highly frequent but none of the moderately frequent items. Likewise, it will not do well on datasets where the correlation structure makes the frequency of items on one level greatly different from that on other levels. Similar conclusions can be drawn for other maximal FIM algorithms.

References

1. Rakesh Agrawal, Tomasz Imielinski, and Arun Swami: Mining Association Rules between Sets of Items in Large Databases. SIGMOD Record, ACM Press (1993) pp 207–216.
2. Rakesh Agrawal and Ramakrishnan Srikant: Fast Algorithms for Mining Association Rules. in Proc. of the 1994 Very Large Data Bases Conference (1994), pp 487–499.
3. Roberto J. Bayardo: Efficiently Mining Long Patterns from Databases. in Proc. of the 1998 ACM SIDMOD Int. Conf. on Management of Data, (1998) pp 85–93.
4. Douglas Burdick, Manuel Calimlim, and Johannes Gehrke: MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. Proc. of the 17th Int. Conf. on Data Engineering (2001) pp 443–452.
5. Nele Dexters, Paul Purdom, Dirk Van Gucht: Analysis of Candidate-Based Frequent itemset Algorithms. Proc. ACM Symposium on Applied Computing, Data Mining Track, (2006) Dijon.
6. Bart Goethals: Frequent Set Mining. The Data Mining and Knowledge Discovery Handbook, Oded Maimon and Lior Rokach (Eds.) (2005), pp 277-397.
7. Karam Gouda and Mohammed J. Zaki. GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets. Data Mining and Knowledge Discovery, 11, (2005) pp 1–20.
8. Jiawei Han, Jian Pei, Yiwen Yin: Mining Frequent Patterns without Candidate Generation. Proc. of the 2000 ACM SIGMOD Int. Conf. Management of Data, Dallas, TX (2000) pp 1–12.
9. Paul W. Purdom, Dirk Van Gucht, and Dennis P. Groth: Average Case Performance of the Apriori Algorithm. SIAM J. Computing, **33**(5),(2004) pp 1223–1260.
10. Mohammed J. Zaki: Scalable Algorithms for Association Mining. IEEE Transactions on Knowledge and Data Engineering, **12**(3), (2000) pp 372–390.

Autonomous Visualization

Khalid El-Arini, Andrew W. Moore, and Ting Liu

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{kbe, awm, tingliu}@cs.cmu.edu

Abstract. Many classification algorithms suffer from a lack of human interpretability. Using such classifiers to solve real world problems often requires blind faith in the given model. In this paper we present a novel approach to classification that takes into account interpretability and visualization of the results. We attempt to efficiently discover the most relevant snapshot of the data, in the form of a two-dimensional scatter plot with easily understandable axes. We then use this plot as the basis for a classification algorithm. Furthermore, we investigate the trade-off between classification accuracy and interpretability by comparing the performance of our classifier on real data with that of several traditional classifiers. Upon evaluating our algorithm on a wide range of canonical data sets we find that, in most cases, it is possible to obtain additional interpretability with little or no loss in classification accuracy.

1 Introduction

In this paper we present a classification algorithm that takes into account human interpretability of the results. We attempt to find the most relevant snapshot of a data set, in the form of a two-dimensional scatter plot with easily understandable axes. This search procedure results in a transformation of our original attributes into two new features, which not only results in a potentially useful visualization of the data, but can be used as the basis for a simple classifier. Using this classifier, we can begin to investigate the trade-off between classification accuracy and interpretability. We call this process *Autonomous Visualization* (AV).

Previously, there has been much work on the visualization of large data sets, which usually involves projecting several dimensions onto a two-dimensional plot that is easy for humans to comprehend. de Oliveira and Levkowitz provide a recent survey of the field [4].

Many have tackled the problem of data-driven scientific discovery [11,6,8]. Others have attempted to add interpretability to clustering [15,14,1]. Our work differs from these in that we are dealing with the problem of classification, and that we do not limit our techniques to any specific application area.

Goldberger et al. describe a dimensionality reduction technique that can be used for producing visualizations of high-dimensional data [9]. While their approach of finding an optimal transformation of the data to a lower-dimensional

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 495–502, 2006.
© Springer-Verlag Berlin Heidelberg 2006

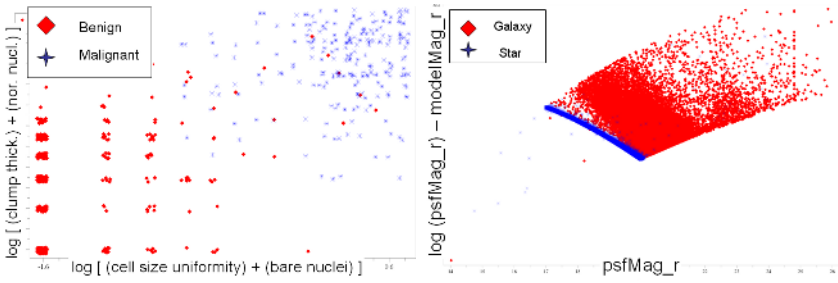


Fig. 1. *Left:* The AV plot for a breast cancer data set separates the benign and malignant cases along a diagonal decision boundary. *Right:* This SDSS star/galaxy plot produced by AV clearly separates the two classes while maintaining interpretable axes.

space before running nearest neighbor is similar in nature to what we are attempting, one key difference is that the scatter plots they produce do not have easily interpretable axes. This is also true for other techniques that produce features that are linear combinations of the inputs, such as principal components analysis and projection pursuit [7,13].

Others have proposed classifiers that take interpretability into account, but we differ by not relying on the format of a rule set or nearest neighbor to make our classifier interpretable, but rather on direct visualization [16,10,3].

2 Methodology

Our approach in designing a visualizable, interpretable classification algorithm assumes that the data consists of real-valued input attributes and a single, discrete output. At a high level, our technique consists of three steps: (1) We search over two-dimensional scatter plots of the data, and select the most relevant plot (where *relevance* in this context is defined below); (2) Given the most relevant plot, we transform the data into the two dimensions defined by its axes, and then train a simple classifier in this transformed space; (3) We classify future data points by transforming them into this two-dimensional space, and applying the classifier trained in the previous step.

2.1 Scatter Plots

In order to find the most relevant snapshot of the data, we search through the space of possible two-dimensional scatter plots. To ensure that the features plotted on the two axes are understandable to humans, as well as to make our search tractable, we limit the types of scatter plots we consider. First, each of the two axes in a scatter plot represents an arithmetic expression of only one or two input attributes. Expressions of more than two attributes begin losing their ease of interpretability. Furthermore, we limit the possible arithmetic expressions to ones that contain commonly understood operators. We refer to the pair of expressions that define a scatter plot as a *pairexp* (e.g., the axes in Figure 1).

2.2 Relevance

The single most important aspect of the Autonomous Visualization process is determining which scatter plots are better than others. This relies on defining a relevance metric that allows us to score pairs of expressions on how likely they are to produce interesting visualizations of the data. One characteristic of visually appealing scatter plots is that points from the same class tend to be grouped together, and apart from points in other classes. We developed a metric based on this intuition, as well as an efficient algorithm to compute these scores.

We consider the input data to have been generated by a set of two-dimensional Gaussian distributions, one per class. Thus, we transform the data into the two dimensions defined by the current pairexp, and compute for each class k the maximum likelihood Gaussian, with mean μ_k and covariance $\mathbf{\Sigma}_k$. We then compute the number of misclassifications that would occur if we used the Gaussians to classify the points in our data set. We define the score of the current pairexp to be this *training set error*. Intuitively, a plot that has well separated classes should obtain a lower training set error, and thus a better score, than a plot whose classes are not well separated. We use training set error rather than validation set error on a held out set because it is precisely the points in the training set that we wish to visualize.

Our score can now be written as $\sum_i I(c_i \neq \hat{c}_i)$, where $I(\cdot)$ is the indicator function, $\hat{c}_i = \text{argmax}_k P(\mathbf{x_i}|c_i = k)P(c_i = k)$ is the class predicted by the Gaussian Bayes classifier, and c_i is the correct output class for point $\mathbf{x_i}$.

2.3 Classification

Once we find the best pair of expressions, we have two human interpretable axes on which to plot our data. However, we also now have transformed the original m-dimensional input space to a two-dimensional space that can be used for Gaussian Bayes classification. We classify new data points using the maximum likelihood Gaussians learned for this new transformed space. Specifically, we compute the predicted class $\hat{c}_i$ for each point to be classified, as above.

3 Acceleration

3.1 Accelerating Score Computation

Naïvely, we can compute the relevance score for a pairexp by iterating through every point in the data set and checking its Gaussian Bayes classification against the true class label. This approach clearly takes linear time in the number of data points, per pairexp. However, by exploiting the spatial structure of our data, we developed an efficient algorithm for computing the score using *kd*-trees [2].

The first step in efficiently computing the score for a given pairexp of up to four attributes (i.e., up to two per expression) is constructing a *kd*-tree with these attributes as its dimensions. Note that although *kd*-tree construction is an $O(n \log n)$ operation, since we are constructing the *kd*-tree with the raw attribute

values (i.e., no expression-specific information) we only need to construct one tree for all possible pairexps of these four attributes. With our current set of legal expressions, this amounts to one tree constructed per nearly 3,000 pairexps.

We traverse the tree linearly, and at a given *kd*-tree node, we attempt to calculate the number of misclassified points belonging to that node without actually iterating through all of them. Given the bounding hyperrectangle for the node, we check to see if any of the class Gaussians are dominant. In other words, for every geometric location $\mathbf{x}$ in the bounding box, does there exist a class k such that $P(c_{\mathbf{x}} = k|\mathbf{x}) > P(c_{\mathbf{x}} = l|\mathbf{x})$ for all classes l? If so, we say that class k dominates this node, since every point will be classified as class k. We can thus prune our search, because all points of other classes belonging to this node will be misclassified, and we can immediately compute the score. If no class dominates the node, we sum the results of recursive calls to the node's two children. We only ever iterate through the individual points of a node if we are at a leaf node that is not dominated by any single class. For more details, please refer to our technical report [5].

Finally, it is important to note that due to the monotonicity of our scoring metric, we can perform *early termination* pruning, which is a significant source of computational gain. As we traverse the *kd*-tree, we can terminate immediately after computing a misclassification score that is greater than the best one so far, even if we have only looked at a tiny portion of the tree. This allows this algorithm to run much faster than the naïve one in practice.

3.2 Accelerating Pairexp Search

Finding the optimal scatter plot that represents the best scoring pairexp involves a hefty combinatorial search. The search can be divided into a two-stage hierarchy. At an outer level, we iterate over tuples of attributes. In an inner loop, given a tuple of attributes, we iterate over possible pairexps that can be generated from those attributes. For high dimensional data sets, it is computationally infeasible for both stages to be exhaustive, since we would have to consider $\mathrm{O}(m^4)$ tuples, where m is the number of dimensions.

Rather than exhaustively considering all possible 2-,3-, and 4-tuples of the input attributes, we instead perform a greedy search, followed by one step of hill-climbing: (1) We exhaustively consider all pairs of attributes, remembering which pair produced the best scoring pairexp (e.g., if the best scoring two attribute pairexp was $[x, \log y]$, then we remember (x, y) as the best pair); (2) Given the best pair, we consider all triples that contain the best pair; (3) Given the best triple, we consider all quadruples that contain the best triple; (4) Starting from the best quadruple, we consider all other quadruples that can be obtained by changing each of the four attributes, one at a time.

Note that only the outer stage is greedy; our inner loop search over pairexps of a given tuple is exhaustive. This heuristic search now only has a quadratic dependency on m, since steps 2-4 are now linear. Furthermore, this quadratic dependency is ameliorated by the fact that, in contrast to quadruples, the number of pairexps that can be generated from a pair of attributes is quite limited.

4 Experimental Results

We evaluated both the naïve and *kd*-tree based implementations on twelve different data sets. Eleven come from the UC Irvine repository, while EDSGC is a subset of the Edinburgh/Durham Southern Galaxy Catalogue.

4.1 Interpretability

We produced a scatter plot for each of the twelve data sets. Figure 1 contains a representative sample, and the entire set of plots is available in our technical report [5]. We see that our algorithm succeeds in producing some separation of the classes. Furthermore, the axes represent simple expressions involving attributes that are well known to the domain experts that would be interested in analyzing the data. This stands in contrast to the alternative of plotting the data as projected onto its top two principal components, since in this case, the axes are no longer directly interpretable.

In order to verify that our algorithm was producing interesting and interpretable visualizations of the data, we consulted a domain expert in astronomy, who provided us with real data from the Sloan Digital Sky Survey (SDSS). Given ten real input attributes, and a 50,000 record subset of the data corresponding to a region of the sky deemed interesting by the astronomer, AV produced the star/galaxy plot seen in Figure 1. The domain expert confirmed that the plot was precisely the kind of plot he would expect from this data. This anecdotal evidence supports our claim that AV indeed produces useful visualizations.

Furthermore, in order to determine whether the complexity of our binary arithmetic expressions was warranted, we ran a version of our algorithm that considered only pairexps with unary expressions on the two axes. When applied to the SDSS data, this version produces a rather uninspiring plot, indicating that there are indeed data sets that are best visualized by combining multiple attributes on a single axis.

4.2 Classification Accuracy

We computed five-fold cross validation accuracies after running our AV classifier on all the data sets. For the sake of comparison, we ran nine canonical classifiers on the same data [17]. It should be noted that, for the sake of expediency, we limited all larger data sets to 5,000 randomly selected records.

When we compare the classification accuracies of our classifier with those of the canonical algorithms, we find that the AV classifier is competitive. In Figure 2, we see a pairwise comparison between our classifier and the nine canonical classifiers. On average, the classification accuracy of AV is only 0.168 percentage points lower than the other algorithms. In fact, AV outperforms 1-nearest neighbor and naïve Bayes. On these data sets, the best performing algorithm was a support vector machine with quadratic kernels, and AV is only 2.348 percentage

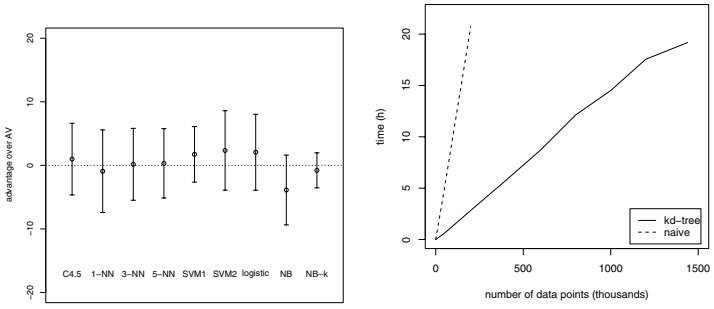


Fig. 2. *Left:* A pairwise comparison between AV and canonical classifiers, showing the difference in classification accuracy, averaged over all data sets. *Right:* Timing analysis of AV on expanded EDSGC data set, as number of records is increased.

points less accurate on average. In Table 1, we see that AV outperforms the *average* canonical classifier on all data sets except iris, realmpg, sonar and vehicle. However, although close at times, AV is never the *most accurate* classifier on a given data set.

4.3 Efficiency

Table 1 shows timing results for running AV with both the *kd*-tree and non-*kd*-tree implementations of the algorithm. We see that by implementing our algorithm using *kd*-trees, we make noticeable gains in efficiency over the naïve implementation, especially for larger data sets. The adult data set, for instance, sees a 4.12 fold speed-up, and the EDSGC data set sees a speed-up of 8.12. It is important to note, however, that the vehicle data set performs slightly more slowly under the *kd*-tree algorithm than the naïve one. This is perhaps due to the relatively small size of the data set preventing the algorithm from overcoming the six pairwise class comparisons it must do per *kd*-tree node, a problem not encountered in the 50,000 record forest data set, despite its seven output classes.

Furthermore, we timed the *kd*-tree implementation on the EDSGC data set as we varied the number of points from 50,000 to the full 1.4 million record data set. The resulting plot, as shown in Figure 2, is mostly linear, but begins to dip slightly as we approach the size of the full data set. When compared to the plot for the naïve implementation, we see that the overhead of building the *kd*-trees is a small price to pay for the gain in efficiency.

It is important to note that the timing costs reported here are only incurred during the training phase of the classifier. Once a relevant pairexp has been found during training, future data points can be classified simply by transforming them to the two-dimensional space and applying the Gaussian Bayes classifier. Alternatively, for a large data set, time can be saved by training only on a subset

Table 1. Classification accuracy and timing results

			Accuracy (%)			Timing (sec)	
Data set	Records	Attributes	best	average	AV	*kd*-tree	Naïve
Abalone	4178	8	55.47	53.28	53.84	308	339
Adult	48844	6	83.92	80.87	82.02	502	2069
Breast-w	701	9	97.42	96.15	96.19	22	59
Diabetes	770	8	77.47	74.51	77.34	22	55
EDSGC	50000	22	99.46	95.16	99.16	2221	18037
Forest	50000	10	69.82	67.40	68.12	4856	8445
Heart-statlog	272	13	85.19	80.16	81.48	21	46
Ionosphere	357	34	91.74	87.59	91.17	69	183
Iris	155	4	96.67	95.78	93.33	2.0	7.2
Realmpg	394	7	82.91	72.79	70.66	2.0	6.8
Sonar	210	60	87.02	77.51	74.04	140	227
Vehicle	851	18	80.50	69.35	60.52	279	268

of the data in order to obtain the best pairexp, but then transforming the entire data set for visualization purposes.

5 Discussion

There is much flexibility inherent to the AV process, which provides ample opportunity for exploration during algorithm design. For instance, before settling on the Gaussian Bayes misclassification score, we implemented several others for comparison. The first attempt was a pairwise count that penalized points of differing classes that were in close proximity to each other. We found that this algorithm generally produced poorer visualizations, and relied on several parameters that we found difficult to set. (Leban et al. propose a similar method, which they use to successfully analyze and visualize gene expression data [12].) In another attempt, we tried maximizing the Gaussian log likelihood of the data, and found that while we produced visualizations that were at least as good as our current results, it was difficult to optimize the algorithm in terms of efficiency, which we believe is the key to making it tractable to search such an intense diversity of scatter plots. We faced a similar situation when we tried a Gaussian misclassification score that allowed full covariance matrices, rather than the diagonal matrices that we use in our current algorithm. In fact, since our relevance score is simply a training set error, conceivably any classifier can be substituted for the Gaussian Bayes classifier that we use, either for computing the score or for classifying new data.

6 Conclusion

We have described a novel approach to classification that takes into account interpretability of the results. We have detailed an algorithm that produces rel-

evant scatter plots of real-valued data sets, and evaluated it in terms of interpretability, classification accuracy, and efficiency. For most problems we considered, our algorithm was competitive with state-of-the-art classifiers. We provided compelling evidence that it is possible to obtain additional interpretability with little or no loss in classification accuracy. Furthermore, we showed that our algorithm is efficient, making it feasible for use on large, real-world data sets. Finally, our results demonstrate the potential for further investigation into the opportunities and challenges of integrating visualization with classification.

References

1. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.
2. J. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.
3. R. Caruana, H. Kangarloo, J. David, N. Dionisio, U. Sinha, and D. Johnson. Case-based explanation of non-case-based learning methods. In *Proc. American Medical Informatics Association Symposium*, pages 212–215, 1999.
4. M.C.F. de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: A survey. *IEEE Trans. Visualization and Computer Graphics*, 9(3), July-September 2003.
5. K. El-Arini, A. W. Moore, and T. Liu. Autonomous visualization. Technical Report CMU-CS-06-137, Carnegie Mellon University, 2006.
6. B. C. Falkenhainer and R. S. Michalski. Integrating quantitative and qualitative discovery: The ABACUS system. *Machine Learning*, 1(4):367–401, 1986.
7. J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Computers*, C-23(9):881–889, 1974.
8. I. Galkin, B. Reinisch, X. Huang, R. Benson, and S. Fung. Automated diagnostics for resonance signature recognition on IMAGE/RPI plasmagrams. *Rad. Sci.*, 2004.
9. J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*. 2005.
10. R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90, 1993.
11. P. Langley. Data-driven discovery of physical laws. *Cog. Science*, 5(1):31–54, 1981.
12. G. Leban, M. Mramor, I. Bratko, and B. Zupan. Simple and effective visual models for gene expression cancer diagnostics. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 167–176, 2005.
13. E. Lee, D. Cook, S. Klinke, and T. Lumley. Projection pursuit for exploratory supervised classification. *Comp. and Graphical Statistics*, 14(4):831–846, 2005.
14. D. Pelleg and A.W. Moore. Mixtures of rectangles: Interpretable soft clustering. In *Proc. 18th International Conference on Machine Learning*, pages 401–408, 2001.
15. T. Sprenger, R. Brunella, and M. Gross. H-BLOB: A hierarchical visual clustering method using implicit surfaces. In *Proc. Visualization '00*, pages 61–68, 2000.
16. S. M. Weiss, R. S. Galen, and P. V. Tadepalli. Maximizing the predictive value of production rules. *Artificial Intelligence*, 45(1-2):47–71, 1990.
17. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, CA, 2000.

Naive Bayes for Text Classification with Unbalanced Classes

Eibe Frank[1] and Remco R. Bouckaert[1,2]

[1] Computer Science Department, University of Waikato, New Zealand
[2] Xtal Mountain Information Technology, Auckland, New Zealand
{eibe, remco}@cs.waikato.ac.nz, rrb@xm.co.nz

Abstract. Multinomial naive Bayes (MNB) is a popular method for document classification due to its computational efficiency and relatively good predictive performance. It has recently been established that predictive performance can be improved further by appropriate data transformations [1,2]. In this paper we present another transformation that is designed to combat a potential problem with the application of MNB to unbalanced datasets. We propose an appropriate correction by adjusting attribute priors. This correction can be implemented as another data normalization step, and we show that it can significantly improve the area under the ROC curve. We also show that the modified version of MNB is very closely related to the simple centroid-based classifier and compare the two methods empirically.

1 Introduction

Multinomial naive Bayes (MNB) is the version of naive Bayes that is commonly used for text categorization problems. In this paper we identify a potential deficiency of MNB in the context of skewed class sizes. The standard practice of initializing word frequencies for all classes to the same value—normally, a value of one is used—biases predictions in favor of the larger class: initial word counts have a larger influence on the predicted probability when there is less data, as in the smaller classes in a text categorization problem. We investigate the use of distinct initial word counts for different-size classes, and propose a heuristic for choosing the initial word count for each class. This modification can be implemented as a pre-processing step that normalizes the word count vector associated with each class and can significantly improve predictive performance as measured by the area under the ROC curve. We also compare the modified version of MNB to the centroid classifier, to which it is closely related.

2 Naive Bayes for Text Classification

In the MNB classifier each document is viewed as a collection of words and the order of words is considered irrelevant. The probability of a class value c given a test document d is computed as

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 503–510, 2006.
© Springer-Verlag Berlin Heidelberg 2006

$$P(c|d) = \frac{P(c)\prod_{w\in d} P(w|c)^{n_{wd}}}{P(d)}, \tag{1}$$

where n_{wd} is the number of times word w occurs in document d, $P(w|c)$ is the probability of observing word w given class c, $P(c)$ is the prior probability of class c, and $P(d)$ is a constant that makes the probabilities for the different classes sum to one. $P(c)$ is estimated by the proportion of training documents pertaining to class c and $P(w|c)$ is estimated as

$$P(w|c) = \frac{1 + \sum_{d\in D_c} n_{wd}}{k + \sum_{w'} \sum_{d\in D_c} n_{w'd}}, \tag{2}$$

where D_c is the collection of all training documents in class c, and k is the size of the vocabulary (i.e. the number of distinct words in all training documents). The additional one in the numerator is the so-called Laplace correction, and corresponds to initializing each word count to one instead of zero. It requires the addition of k in the denominator to obtain a probability distribution that sums to one. This kind of correction is necessary because of the zero-frequency problem: a single word in test document d that does not occur in any training document pertaining to a particular category c will otherwise render $P(c|d)$ zero.

3 Unbalanced Class Sizes

Many text categorization problems are unbalanced. This can cause problems because of the Laplace correction used in (2). Consider a word w in a two-class problem with classes c_1 and c_2, where w is completely irrelevant to the classification. This means the odds ratio for that particular word should be one, i.e. $\frac{p(w|c_1)}{p(w|c_2)} = 1$, so that this word does not influence the class probability.

Assume the word occurs with equal relative frequency 0.1 in the text of each of the two classes. Assume there are 20,000 words in the vocabulary ($k = 20,000$) and the total size of the corpus is 100,000 words. Figure 1 shows the estimated odds ratio, based on (2), as the relative size of the two classes changes. It has the desired value of one when both classes contain the same number of words. However, the situation changes dramatically as the class sizes become skewed. For example, when c_2 becomes very small and contains little text relative to the other class, the presence of the irrelevant word w in a test document substantially increases the estimated probability that the test document belongs to class c_1.

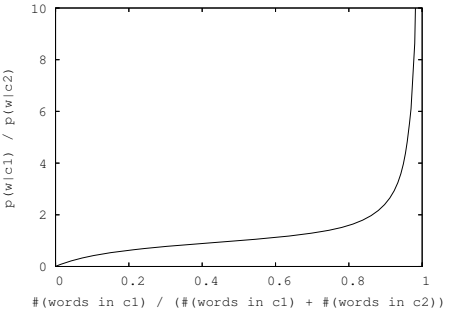


Fig. 1. Estimated odds ratio

Fortunately it turns out that there is a simple remedy: we can normalize the word counts in each class so that the total size of the classes is the same for both classes after normalization. To do this we replace n_{wd}, where d is in D_c, by

$$\alpha \times \frac{n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}}, \tag{3}$$

i.e. we normalize the vector of word counts for each class to have length α when measured according to L_1 norm. This ensures that the estimated odds ratio for our irrelevant word will be one, regardless of which particular value we choose for the normalization constant α.

We can also view this normalization step as a modification of the Laplace correction. Plugging the normalized counts into (2), it is easy to show that we have effectively replaced the standard initial word count of one by the class-specific initial word count $(\sum_{w'} \sum_{d \in D_c} n_{w'd})/\alpha$. This means that we are using asymmetric word count priors.

The value of α determines the amount of smoothing across word counts in the dictionary. Surprisingly, initial experiments on the Reuters data showed that a value of $\alpha = 1$ often works well, which corresponds to very heavy smoothing. As we will see, $\alpha = 1$ also results in good performance on other datasets included in our experimental comparison. In the next section we will show that our modified MNB with $\alpha = 1$ is closely related to the centroid classifier for text classification.

4 Relationship to Centroid Classifier

In the centroid classifier [3], each class is represented by its mean word vector, normalized to unit length using L_2 norm. The centroid $\mathbf{c}_c$ for class c is given by

$$\mathbf{c}_c = \{\frac{\sum_{d \in D_c} n_{w_1 d}}{\sqrt{\sum_w (\sum_{d \in D_c} n_{wd})^2}}, \frac{\sum_{d \in D_c} n_{w_2 d}}{\sqrt{\sum_w (\sum_{d \in D_c} n_{wd})^2}}, \ldots, \frac{\sum_{d \in D_c} n_{w_k d}}{\sqrt{\sum_w (\sum_{d \in D_c} n_{wd})^2}}\}.$$

Assuming $\mathbf{x}_d = \{n_{w_1 d}, n_{w_2 d}, \ldots, n_{w_k d}\}$ is the word vector representing a test document d, the scoring function for the centroid classifier is

$$\mathbf{x}_d \cdot \mathbf{c}_1 - \mathbf{x}_d \cdot \mathbf{c}_2. \tag{4}$$

Now consider MNB. The scoring function (log odds) for MNB can be written as

$$\left[\log P(c_1) + \sum_{i=1}^{k} n_{w_i d} \log(P(w_i|c_1))\right] - \left[\log P(c_2) + \sum_{i=1}^{k} n_{w_i d} \log(P(w_i|c_2))\right]. \tag{5}$$

The terms $\log P(c_1)$ and $\log P(c_2)$ are irrelevant when we rank documents. Comparing (4) and (5) we see that the only remaining difference is that $\log(P(w_i|c))$ is used instead of the corresponding vector component from the centroid. However,

these two terms have a very similar effect if we set α to one in our modified version of MNB. When $\alpha = 1$, using (3) in (2) means that

$$P(w|c) = \frac{1 + \frac{\sum_{d \in D_c} n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}}}{k+1}.$$

The denominator is constant and can be dropped. Furthermore,

$$\frac{\sum_{d \in D_c} n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}} << 1$$

for practical text datasets, and $\log(1 + x) \approx x$ for $x << 1$. Hence the only remaining difference between the scoring functions in (4) and (5) is that L_2 norm is used in the former and L_1 norm in the latter.

As we will see in the next section, the modified MNB classifier with $\alpha = 1$ indeed gives very similar results to the centroid classifier on most of the datasets we investigated in our experiments.

5 Empirical Results

In this section we present experiments comparing MNB with and without our modification, and the centroid classifier. Weka was used for the experiments, and the area under the ROC curve (AUC) employed as the performance statistic. All results are averages from ten runs of the hold-out method. For each run 66% of the data was used for training and 34% for testing, with stratification to ensure that class proportions were preserved. The same runs were used for each of the learning schemes that we investigated. To test for significant differences we used the corrected resampled paired t-test [4], which has acceptable Type I error.

Our experiments were based on four well-known text classification datasets: `Reuters-21578`, `WebKB`, `Industry Sector`, and `20 Newsgroups`. For each dataset we created as many two-class classification problems as there were class values in the data, with the exception of the `Reuters-21578` data where we only used the 10 most frequent categories. All documents pertaining to a particular class value were put into one category and all the remaining documents in the other. This was necessary in order to use AUC for evaluation. Consequently we created 10 classification problems from the `Reuters-21578` data, four from the `WebKB` data, 105 from the `Industry Sector` data, and 20 from the `20 Newsgroups` data.

For each dataset we used the same steps to extract word features. All characters were converted to lowercase, only alphabetic tokens were considered, stopwords and *hapax legomena* were removed, and the *full* resulting vocabulary was used (i.e. no feature selection was performed).

We applied the following pre-processing steps to the raw word counts [1,2]. First, TF×IDF was used to transform n_{wd} into $n'_{wd} = \log(1+n_{wd}) \times \log(m/m_w)$, where m is the total number of training documents and m_w is the number of training documents that contain w. Secondly, following this transformation, the

vector of transformed word frequencies for each document was normalized to length l using L_2 norm to counteract the effect of varying document lengths. The vector length l was set to the average vector length in the training documents before normalization. The transformed and normalized frequencies were then used in both versions of MNB and the centroid classifier.

We will first look at the impact of our proposed modification using $\alpha = 1$, which is equivalent to normalizing the word vector for each class to length one in L_1 norm before applying standard MNB. Figure 2 shows the results for both standard MNB and MNB with per-class normalization (MNB_{PCN}). Bars that are striped mark differences that are statistically significant at the 5% level (in these figures and all other figures in this paper).

The results show that our modification significantly improves the performance of MNB on three of the ten `Reuters-21578` categories. It significantly reduces AUC on two categories. However, not taking the significance of the individual differences into account, there are eight wins and only two losses. This win/loss ratio has a p-value of 0.11 according to a two-sided sign test.

On the `Web KB` data the results are clear cut. MNB_{PCN} achieves significant gains on all four categories. Although there is only one significant win on the `Industry Sector` data, there is no significant loss. Moreover, not taking significance on individual `Industry Sector` categories into account, the win/loss ratio is 81/24 in favor of the modified version, which is highly significant according to a two-sided sign test (p-value $= 4.64 \times 10^{-8}$).

The `20 Newsgroups` data is the outlier in this collection of results. Here MNB outperforms MNB_{PCN}: there are 18 significant wins for MNB, and no significant losses (without considering significance, the win/loss ratio is 20/0). However, it turns out that the performance of MNB_{PCN} can be improved by using a different value for α when normalizing the word vectors for each class. Figure 3 compares MNB to MNB with per-class normalization when the length of the word vector for each class (i.e. α) is set to the minimum of the two class vector lengths before normalization (MNB_{PCNmin}). Using this value of α gives a much lower influence to the attribute priors, and the results show that this is beneficial. MNB_{PCNmin} is on par with standard MNB on the `20 Newsgroups` categories: there is one significant win and one significant loss. The win/loss ratio is 10/10 when significance of individual differences is not taken into account. Hence it appears that less smoothing is beneficial for the `20 Newsgroups` data.

Unfortunately the new value for α is no silver bullet. Although it improves performance further on the `Industry Sector` data, it results in significant degradation on `Reuters-21578` and `WebKB`.[1] Hence the right amount of smoothing for $P(w|c)$ depends on the domain, and, for best performance, an appropriate value of α should be chosen using a validation set or cross-validation.

In Section 4 we have shown that MNB with per-class normalization and $\alpha = 1$ is closely related to the centroid classifier. In the following we investigate this relationship empirically. Figure 4 shows the AUC for the centroid classifier and the modified version of MNB using $\alpha = 1$.

[1] These results are not included here due to space constraints.

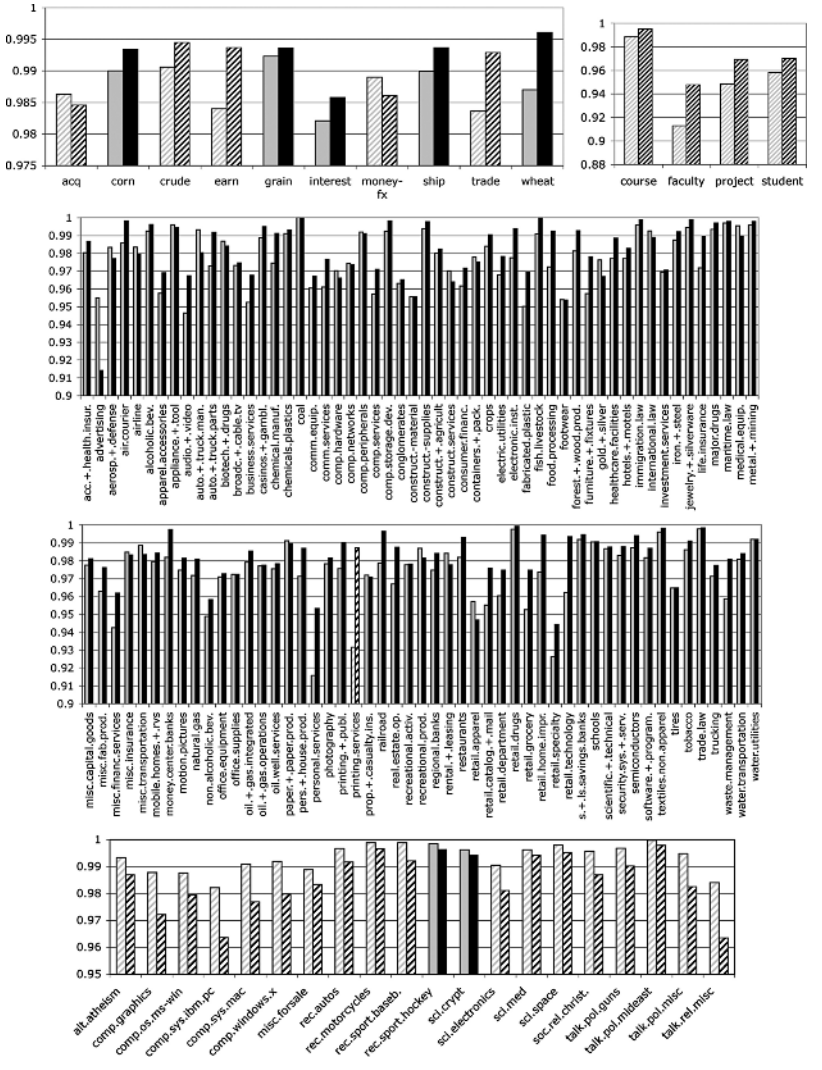


Fig. 2. AUC for MNB (grey) and MNB$_{PCN}$ (black) on Reuters (top left), WebKB (top right), Industry Sector (two middle graphs), and 20 Newsgroups (bottom)

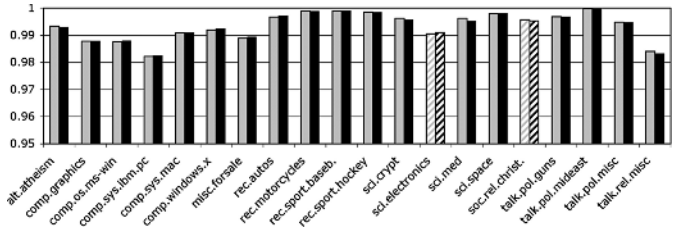


Fig. 3. AUC for MNB (grey) and MNB$_{PCNmin}$ (black) on 20 Newsgroups

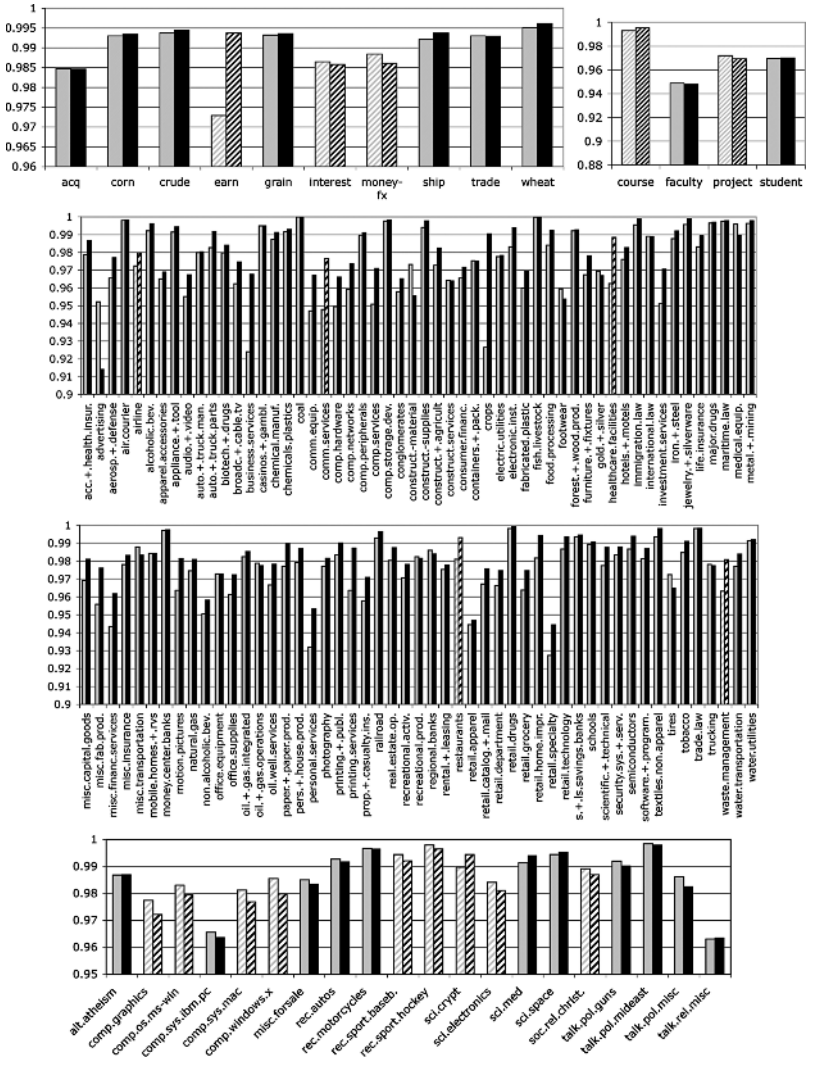


Fig. 4. AUC for centroid classifier (grey) and MNB$_{PCN}$ (black) on Reuters (top left), WebKB (top right), Industry Sector (two middle graphs), and 20 Newsgroups (bottom)

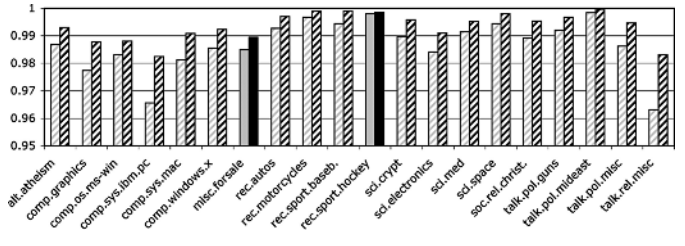


Fig. 5. AUC for centroid classifier (grey) and MNB$_{PCNmin}$ (black) on 20 Newsgroups

Performance of the two methods is indeed very similar in most cases. Ignoring the magnitude of the differences there is no clear winner on the `Reuters-21578` data (two significant wins and one significant loss for the centroid classifier) and no clear winner on the `Web KB` data (one significant win and one significant loss).

However, MNB_{PCN} has an edge on the 105 `Industry Sector` categories. There are five significant losses for the centroid classifier and no significant win, and it almost always yields a lower AUC value. On the other hand, the centroid classifier performs better than MNB_{PCN} on the `20 Newsgroups` data, with eight significant wins and only one significant loss.

Although the AUC of MNB_{PCN} and the centroid classifier is often very similar, there are some noticeable differences, e.g. on the category `earn` in the `Reuters-21578` data. The obvious question is which of the differences discussed in Section 4 is responsible for these discrepancies. To this end we performed a further experiment where L_1 normalization was used to normalize the centroids in the centroid classifier. The AUC scores for this modified centroid classifier were virtually indistinguishable from those obtained using MNB_{PCN}: they did not differ at all when rounded to the fourth decimal place. Hence the differences in Figure 4 are almost exclusively due to L_1 normalization vs. L_2 normalization.

Note that MNB with per-class normalization outperforms the standard centroid classifier with L_2 normalization if α is set to the minimum vector length before normalization (as in Figure 3). This is shown in Figure 5. There are eighteen significant wins for MNB and no significant losses (the win/loss ratio is 20/0 if significance is not taken into account). A similar win/loss ratio holds for standard MNB, which also outperforms the centroid classifier on this data.

6 Conclusions

In this paper we have identified a potential deficiency of MNB in the context of unbalanced datasets and shown that per-class word vector normalization presents a way to address the problem. Our empirical results show that normalization can indeed significantly improve performance. We have also shown that MNB with class vector normalization is very closely related to the standard centroid classifier for text classification if the class vectors are normalized to unit length, and verified the relationship empirically.

References

1. Rennie, J.D., Shih, L., Teevan, J., Karger, D.R.: Tackling the poor assumptions of naive Bayes text classifiers. In: Proc Int Conf on Machine Learning. (2003) 616–623
2. Kibriya, A.M., Frank, E., Pfahringer, B., Holmes, G.: Multinomial naive Bayes for text categorization revisited. In: Proc Australian Conf on AI. (2004) 488–499
3. Han, E.H., Karypis, G.: Centroid-based document classification: Analysis and experimental results. In: Proc European Conf on Principles and Practice of Knowledge Discovery in Databases. (2000) 424–431
4. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning **52**(3) (2003) 239–281

Knowledge-Conscious Data Clustering

Amol Ghoting and Srinivasan Parthasarathy

The Ohio State University, Columbus, OH 43210, USA
{ghoting, srini}@cse.ohio-state.edu

Abstract. We consider the problem of efficiently executing data clustering queries in a client-server setting. Extant solutions to this problem suffer from (a) a significant amount of remote I/O and (b) minimal re-use of computation between both iterations of a kMeans query, and executions of different kMeans queries. We propose to facilitate interactive kMeans clustering by employing a *client-side knowledge-cache*. This knowledge-cache is succinct and significantly reduces the amount of remote I/O needed during execution. Furthermore, it permits the re-use of computation, both within and between executions of the kMeans queries.

1 Introduction

The knowledge discovery process is interactive in nature. Therefore, minimizing query response-time is imperative, because a lengthy delay can disrupt the formation of insight. To address this challenge, the past few years have seen researchers make significant progress in reducing the time required to execute a single data mining query. However, given the iterative and interactive nature of the knowledge discovery process, one expects there to be significant repeated computation through successive executions of a data mining algorithm. Therefore, an orthogonal and potentially beneficial approach to reduce a query's response-time would be to expose redundant computation between executions, cache this computation, and re-use this cached computation in successive executions of the algorithm. While the database community has looked at employing such a *knowledge-conscious* approach to improve query processing performance, such efforts are largely in their infancy in the data mining community [3,4,5].

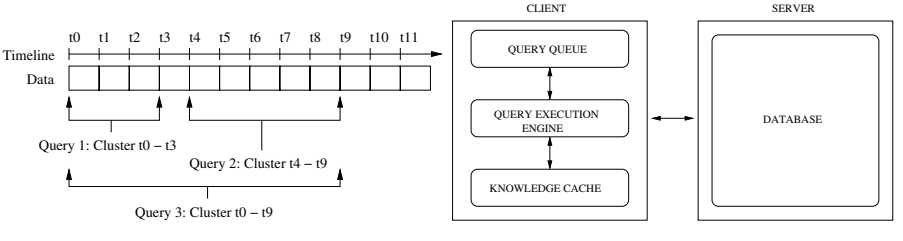


Fig. 1. a)Exploratory data clustering b)Knowledge-conscious mining framework

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 511–519, 2006.
© Springer-Verlag Berlin Heidelberg 2006

The architecture of a knowledge-conscious mining framework is presented in Figure 1b. The system consists of a *client* and a *server*. The server maintains a *database*, while the client manages a *query queue*, a *query execution engine*, and a *knowledge cache*. The query execution engine accepts a query from the query queue, and executes the query using the contents of the (local) knowledge cache and the (remote) database. Furthermore, using the information gathered through an execution, the query execution engine updates the contents of the knowledge cache to improve performance when answering future queries.

In this paper, we consider the problem of efficiently executing exploratory data clustering queries [1] in which the user is interested in interactively clustering different subsets of a data set to study its evolving behavior (Figure 1a). Existing solutions to this problem suffer from (a) a significant amount of remote I/O during execution and (b) a substantial amount of repeated computation through iterations of a kMeans query and multiple kMeans queries. We show how exploratory kMeans clustering can be made knowledge-conscious using a *client-side knowledge cache*. This cache significantly reduces the amount of remote I/O needed during execution. Furthermore, it also reduces the redundant computation between both iterations of a kMeans query, and executions of different kMeans queries.

2 Background

Given a data set D consisting of n data points, each with d dimensions, the data clustering problem is to partition this data set into k subsets such that each subset behaves "well" under some measure. The popular kMeans clustering algorithm can be briefly described as follows. First, it begins with k random centers, $C^0 = \{C_1^0, \cdots, C_k^0\}$. Next, for each of the n data points, it finds its closest center in C^0. The data points are partitioned into k subsets based on their closest centers. The center of mass for each of these k subsets is used to find the new set of k centers, $C^1 = \{C_1^1, \cdots, C_k^1\}$. This process continues iteratively until we encounter an iteration i such that the centers C^i and $C^{(i+1)}$ are identical. Each iteration of this naive kMeans algorithm scales as $O(nkd)$.

The state-of-the-art kMeans clustering algorithm, due to Pelleg and Moore, improves the performance of the above mentioned algorithm by employing a *multi-resolution kd-tree* [6]. Multi-resolution kd-trees have the following properties. First, they are binary trees. Second, each node in the kd-tree contains information about all points contained in a hyper-rectangle h. This hyper-rectangle is stored at the node using two boundary vectors h^{max} and h^{min}. At each node is also stored the *number* and the *center of mass* of all points that lie within h. All the children of a node represent hyper-rectangles contained within h. Third, each node has a *split dimension* and a *split point* assigned to it. The value of the split point on the split dimension is referred to as the *split value* of the node. The children of a node represent two hyper-rectangles such that all points with

values less than the split value on the split dimension are assigned to one child, and all points with values greater than the split value on the split dimension are assigned to the other child. This data structure has exactly n nodes.

Given a set of centers C^i and a hyper-rectangle h, $owner(h, C^i)$ is defined as the center $c \in C^i$ for which any point in h is closer to c than any other center in C^i. Note that h does not always have an owner in C^i. Pelleg and Moore used this concept of ownership to improve the performance of the kMeans algorithm. The multi-resolution kd-tree is used to assign the points to the k centers in each iteration. The algorithm proceeds recursively and can be briefly described as follows. First, beginning with the root node, it checks to see if the hyper-rectangle associated with the node has a unique owner in C^i. If we have a unique owner, statistics stored at the node (*number of points* and *center of mass*) can be used to assign all points covered by the hyper-rectangle to the unique owner, and the procedure can then return. Otherwise, the split point associated with the node is assigned to one of the k centers, and the children of the node are processed in a similar fashion, recursively. In the worst case, we need $O(k^2d)$ operations to process each node. Therefore, if the entire kd-tree needs to be processed, this algorithm will incur significant overheads from ownership checks and each iteration will scale as $O(nk^2d)$. The authors show that on most data sets, hyper-rectangles with unique owners can be discovered early in the search (i.e. at high levels of the kd-tree), affording a significant performance improvement.

In exploratory data clustering [1], as can be seen in Figure 1a, the user is interested in interactively clustering different subsets of the data set D. Furthermore, during this process, the user is also interested in varying k, the desired number of clusters. Such an exploration of the data can provide the user with a much deeper understanding of the evolving behavior of the clusters [1]. In order to perform exploratory data clustering using the state-of-the-art, a system typically proceeds as follows. First, it queries the remote database to retrieve the desired subset of the data. Next, it builds a multi-resolution kd-tree using the data retrieved from the database. Finally, it uses the aforementioned variant of the kMeans algorithm (due to Pelleg and Moore) to cluster the data.

3 Algorithmic Improvements

Existing solutions to exploratory data clustering suffer from the following drawbacks. First, when the database it very large and cannot be cached on the client's side, during query execution, the system needs to retrieve a significant amount of data from the remote database. This is often time-consuming. Second, there is no re-use of computation between iterations of the kMeans algorithm and between executions of two different kMeans clustering queries. This redundant computation is often excessive and significantly affects performance. We propose to facilitate exploratory data clustering by employing a *client-side knowledge cache* to tackle the above mentioned challenges.

Reducing Remote I/O: In order to reduce remote I/O during execution, we propose to maintain a *low-resolution summary* of the data set on the client's side.

This low-resolution summary must have the following characteristics. First, given that a summary with a satisfactory resolution is available, a kMeans clustering using this summary should be identical to that when using the entire data set. Second, when the clustering cannot be performed accurately, we should be able to improve the resolution of this summary to the desired level, incrementally, accessing only a small subset of the remote database.

In an exploratory setting, a data clustering query takes the following form: $Cluster(t_s, t_e, k)$, where t_s and t_e represent the start and the end of the desired range, and k represents the desired number of clusters. To create a client-side knowledge cache, first, we propose to divide the entire data set D into blocks, $D_1, D_2, ..., D_m$, as can be seen in Figure 2. This partitioning is only conceptual and does not need to be physically realized. An execution of a kMeans clustering query typically builds a single kd-tree for all the points between t_s and t_e and iteratively assigns the points and hyper-rectangles in this kd-tree to the k centers. Instead of building a single kd-tree for the entire range, we propose to build a kd-tree for each of the blocks in D, as and when these data blocks intersect the range specified in a kMeans query. This set of kd-trees contains all the information that would be otherwise needed when clustering using a single kd-tree. Such a partitioning allows us to define a unit of re-use between queries spanning different ranges. Second, after query execution, for each kd-tree that is built, the sub-tree that is accessed when executing the kMeans query is stored at the client. This *cached sub-tree* is a low-resolution summary of the data in a block. Furthermore, as the kd-tree is a hierarchically defined structure, this cached sub-tree is also a complete summary of the data in a block. In other words, every data point in the block is either represented as a point or is covered by a hyper-rectangle in this sub-tree. Therefore, in most situations these cached sub-trees on the client's side can be used to perform kMeans clustering without ever contacting the server, significantly reducing remote I/O.

When executing a query, we may encounter a leaf node in the cached sub-tree that does not have a single owner. This situation requires that we increase the resolution of this cached sub-tree. To do so, we need to re-construct the portion of the kd-tree below this leaf node in the cached sub-tree. However, as it stands this sub-tree cannot be grown incrementally. To facilitate incremental growth, we re-order points in each data block as per the points order in the depth-first traversal of the kd-tree for the block. Consequently, when we need to expand a node in the cached sub-tree, all the data points needed to re-construct the portion of the kd-tree below a node will be stored sequentially on the server (Figure 2 illustrates how all children of node number 8 are stored sequentially in the database after depth-first re-ordering). The data points required for node expansion can be identified by simply using a starting and an ending position in the database, and can be retrieved efficiently. We would like the readers to note that using cached sub-trees does not affect the correctness of the kMeans algorithm. We will find the same set of centers as the naive kMeans algorithm.

Reducing the Number of Ownership Checks: Ownership checks are expensive; they need $O(k^2)$ time per node in the cached sub-tree in the worst case.

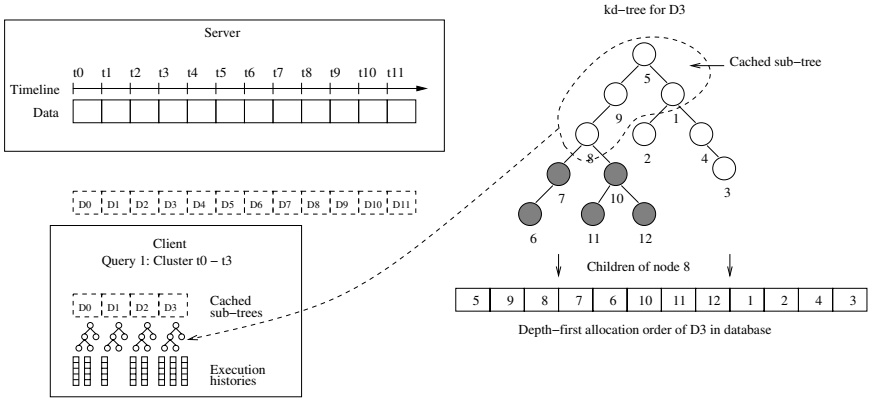


Fig. 2. Client-Side Knowledge Caching

The nodes in the cached sub-tree that are close to the root of the tree tend to encode low-resolution information. This resolution progressively increases as we descend to the lower levels of the cached sub-tree. Therefore, in the cached sub-tree, ownership checks are likely to fail (or not identify a unique owner) at the higher levels and be successful (or identify a unique owner) at some intermediate levels. Furthermore, for two sets of centers, C^i and C^j, that are very similar, during execution, failures and successes in ownership checks when processing a cached sub-tree are likely to be aligned. To benefit from this behavior, we encode the execution history of an iteration of the kMeans query in the knowledge cache. This execution history tracks whether the ownership checks succeeded or failed for each node in the cached sub-tree. This execution history is stored for a set of centers for each cached sub-tree. Therefore, each cached sub-tree can have multiple execution histories, one for each set of distinct centers encountered. When executing an iteration of the kMeans query, for the set of centers to be used in the iteration, we check to see if we have an execution history for a similar set of centers in the knowledge cache. If such an execution history is available, we use it to skip owner checks that are likely to fail. This drastically reduces the number of failing ownership checks and can significantly speed up execution. We would like to point out that skipping an ownership check that would in fact succeed does not affect the correctness of the algorithm.

Reducing the Number of Candidate Centers between Iterations: When executing a kMeans query, the main operation is that of assigning a point or a hyper-rectangle to one of the k candidate centers. In order to make an assignment, for a data point, we need $O(k)$ computations, and for a hyper-rectangle, we need $O(k^2)$ computations. Using the execution history of a query, we propose to reduce the set of candidate centers, and thereby reduce the number of computations needed to make an assignment.

Let us assume that in iteration i of a kMeans query, data points and hyper-rectangles in the cached sub-tree are assigned to one of k centers in C^i. Let $C^{(i+1)}$ be the new set of k centers to be used in $(i+1)^{th}$ iteration. Let $rad(C^i_j)$ be the

radius of the j^{th} center in C^i. Let $d(C^i_j, C^i_k)$ be the euclidean distance between centers C^i_j and C^i_k. Let $maxdist(C^i_j, h)$ be the maximum distance between h (a hyper-rectangle or a point) and a center C^i_j.

Lemma 1. *If a hyper-rectangle or a data point is assigned to center C^i_j in iteration i, then in the $(i+1)^{th}$ iteration, it cannot be assigned to any center $C^{(i+1)}_k$ for which $d(C^i_j, C^{(i+1)}_j) + d(C^i_k, C^{(i+1)}_k) < d(C^i_j, C^i_k)/2 - rad(C^i_j)$.*

Lemma 2. *If a hyper-rectangle (or a data point) h is assigned to center C^i_j in iteration i, then in the $(i+1)^{th}$ iteration, it cannot be assigned to any center $C^{(i+1)}_k$ for which $d(C^i_j, C^{(i+1)}_j) + d(C^i_k, C^{(i+1)}_k) < d(C^i_j, C^i_k)/2 - maxdist(C^i_j, h)$.*

Lemma 3. *Let C^i_j be the j^{th} center in iteration i and $C^i_{j'}$ be the center that is closest to C^i_j. For a hyper-rectangle (or data point) h, if $d(C^i_j, C^i_{j'})/2 > maxdist(C^i_j, h)$, then C^i_j is the unique owner of h.*

The kMeans using kd-trees algorithm is very easily modified to benefit from the aforementioned lemmas. We augment the execution history stored in the knowledge-cache to maintain *radius* of each center at the end of an iteration, and *maxdist* and *assigned center* for each data point and hyper-rectangle in the cached sub-tree. During the iterations of the kMeans algorithm, most data points do not change assignments. For a data point or hyper-rectangle that is assigned to a center C^i_j in iteration i, Lemma 1 allows us to prune away centers that are not candidates in iteration $i+1$, even before we start processing the cached sub-tree. This reduced set of candidate centers is reduced even further using Lemma 2 that considers *maxdist* for every data point or hyper-rectangle encountered. Consequently, using Lemmas 1 and 2, data points or hyper-rectangles that are not likely to be assigned to a new center can be processed in $O(1)$ number of floating point computations. In addition, for data points or hyper-rectangles that are likely to change assignments, the pruned set of candidate centers due to Lemmas 1 and 2 significantly improves performance. When a data point or hyper-rectangle has multiple candidate centers, Lemma 3 is used as an initial check, and is some cases, can help make assignments in $O(1)$ number of floating point computations (please refer to [2] for further details on the lemmas).

Reducing Redundant Computations between kMeans Queries: When the system executes the first kMeans query, during the first iteration, the cached sub-trees are allocated. Through subsequent iterations, old execution histories are used or new execution histories are created and saved. For a subsequent kMeans query, as discussed, one can re-use the cached sub-trees from a previous kMeans query. Furthermore, one can also re-use old execution histories. To understand how execution histories can be used between queries, to begin with, let us assume that the number of desired centers k' for the new kMeans query q' is the same as k[1], the number of centers desired in the previous query q whose

[1] We can also reuse the cached execution history of a query q when the new query q requires k clusters, which is different from k. Please refer to [2] for further details.

execution history has been cached. The key to understanding how execution histories can be re-used lies in the fact that Lemmas 1 and 2 are in fact iteration independent. In other words, the execution history for an iteration i can be used for any iteration (say $i + 5$), not just iteration $i + 1$. Consequently, when the new query presents us with k' initial random centers, we simply postulate that these initial random centers were produced at the end of some iteration when processing query q. Therefore, the execution history for some iteration i in q's execution can be used to speed up the execution of this new query q'. However, to do so, we need to establish a 1-to-1 correspondence between the k cached centers and the k' new random centers. Note that any 1-to-1 correspondence will suffice. We use a simple heuristic in which for each cached center, we locate its closest center amongst the k' centers. To breakup ties, once a new random center has been deemed to be the closest center for some cached center, we do not consider it when finding the closest centers for the other cached centers.

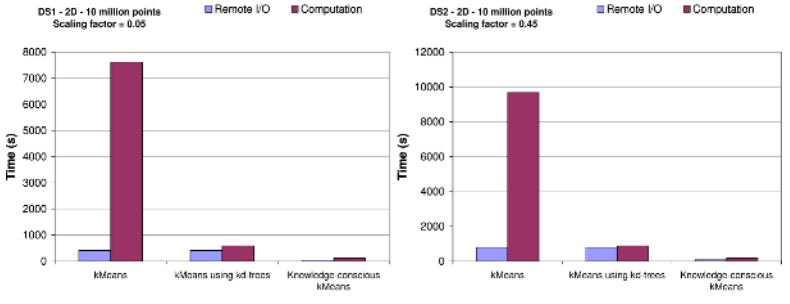


Fig. 3. Reduction in remote I/O and computation - DS1 and DS2

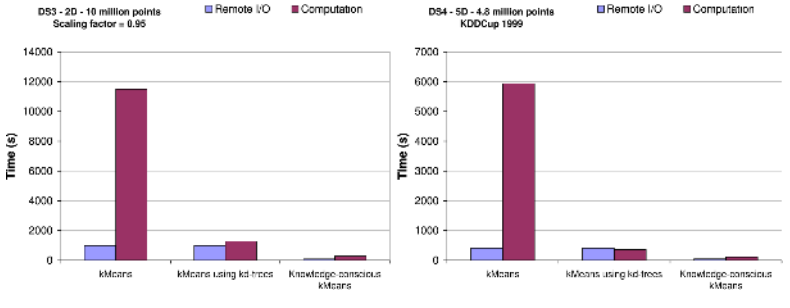


Fig. 4. Reduction in remote I/O and computation - DS3 and DS4

4 Experimental Results

In this section, we will evaluate the performance benefits of our optimizations on a variety of data sets. We use two nodes in an Intel Pentium 4-based cluster, and use MPI for message passing. We consider both synthetic and real data sets for our performance evaluation (please see [2] for details). For the synthetic data

sets, we scale each cluster using a *scaling factor* to vary the data set characteristics. A scaling factor of 0.05 represents a data set with well separated clusters, while a scaling factor of 0.95 represents a data set with clusters that overlap. The real data set we consider is the kddcup 1999 intrusion detection data set. We construct a synthetic kMeans query workload consisting of 30 queries for our experiments; we are not aware of any real clustering workload. The desired number of clusters (between 0 and 100) and the desired range for each query in the workload are set randomly.

Figures 3 and 4 show the time required for remote I/O and computation for the naive kMeans, the kMeans using kd-trees, and the knowledge-conscious kMeans algorithms. DS1 through DS3 are synthetic data sets with varying scaling factors, while DS4 is the kddcup data set. On these four data sets, we see up to a 10-fold reduction in remote I/O time for the knowledge-conscious kMeans algorithm when compared with the naive kMeans and the kMeans using kd-trees algorithms. Furthermore, we see up to a 6-fold reduction in computation due to knowledge re-use for the knowledge-conscious kMeans algorithm when compared with the kMeans using kd-trees algorithm. This represents an overall 8-fold reduction in execution time for the knowledge-conscious kMeans algorithm when compared with the kMeans using kd-trees algorithm. Furthermore, the reduction in computation (up to 6-fold) is observed even when the entire remote database can be cached locally. This experiment serves to illustrate that both cached sub-trees and computation re-use can significantly improve the performance of exploratory kMeans queries.

5 Conclusion

In this paper, we considered the problem of efficiently executing exploratory kMeans clustering queries in a client-server setting. Extant solutions to this problem suffer from a significant amount of remote I/O and repeated computation during execution. To address this challenge, we proposed to use a client-side knowledge-cache with cached-subtrees that provide both compact and complete representations of the remote data set. We also proposed to maintain execution histories in this knowledge-cache to help reduce redundant computation between both iterations of a kMeans query, and multiple kMeans queries. These optimizations afford nearly an order of magnitude reduction in execution time.

Acknowledgments. This work is supported in part by NSF grants #CAREER-IIS-0347662, #RI-CNS-0403342, and #NGS-CNS-0406386.

References

1. C. Aggarwal et al. A framework for clustering evolving data streams. *VLDB'03*.
2. A. Ghoting et al. Knowledge-conscious data clustering. OSU-CISRC-7/06-TR65.
3. D. Judd et al. Large scale parallel data clustering. *ICPR'96*.

4. B. Nag et al. Using a knowledge cache for interactive discovery of association rules. *SIGKDD'99*.
5. S. Parthasarathy and S. Dwarkadas. Shared state for distributed interactive data mining applications. *JPDD'02*.
6. D. Pelleg and A. Moore. Accelerating exact kmeans algorithms with geometric reasoning. *SIGKDD'99*.

On the Lower Bound of Reconstruction Error for Spectral Filtering Based Privacy Preserving Data Mining

Songtao Guo[1], Xintao Wu[1,⋆], and Yingjiu Li[2]

[1] UNC Charlotte
[2] Singapore Management Univ.

Abstract. Additive Randomization has been a primary tool to hide sensitive private information during privacy preserving data mining. The previous work based on Spectral Filtering empirically showed that individual data can be separated from the perturbed one and as a result privacy can be seriously compromised. Our previous work initiated the theoretical study on how the estimation error varies with the noise and gave an upper bound for the Frobenius norm of reconstruction error using matrix perturbation theory. In this paper, we propose one Singular Value Decomposition (SVD) based reconstruction method and derive a lower bound for the reconstruction error. We then prove the equivalence between the Spectral Filtering based approach and the proposed SVD approach and as a result the achieved lower bound can also be considered as the lower bound of the Spectral Filtering based approach.

1 Introduction

Additive randomization has been a primary tool to hide sensitive private information during privacy preserving data mining. Consider a data set U with m records of n attributes and a noise data set V with same dimensions as U. The random value perturbation techniques generate a perturbed data matrix $\tilde{U} = U + V$. Let $\hat{U}$ denote the estimate which the users (or attackers) can achieve. To preserve utility, we mean certain aggregate characteristics (i.e., mean and covariance matrices for numerical data, or marginal totals in contingency table for categorical data) of U should remain basically unchanged in perturbed data $\tilde{U}$ or can be restored from estimated data $\hat{U}$. In other words, distributions of U can be approximately reconstructed from perturbed data $\tilde{U}$ when some a-priori knowledge (e.g., distribution, statistics etc.) about noise V is available using distribution reconstruction approaches (e.g., [2,1]).

To preserve privacy, we mean not only the difference between $\tilde{U}$ and U but also that of between $\hat{U}$ and U should be larger than some tolerated threshold. The authors, in [5] proposed a random matrix-based spectral filtering technique to retrieve original individual data from the perturbed data (a similar Principle Component Analysis based approach investigated in [4]). As a result individual privacy can be seriously compromised.

The previous work only empirically assessed the effects of perturbation on the accuracy of the estimated individual value. Hence one important question is what the

⋆ This research was supported by USA National Science Foundation Grant CCR-0310974 and IIS-0546027.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 520–527, 2006.
© Springer-Verlag Berlin Heidelberg 2006

explicit form between reconstruction accuracy and noise added. In our recent work [3], we derived one upper bound for the Frobenius norm of $\hat{U} - U$ in terms of some knowledge of V (e.g., the Frobenius and 2-norm of V) using the matrix perturbation theory. In this paper, we focus on the lower bound for the Frobenius norm of $\hat{U} - U$. While the upper bound may be exploited by attackers to determine how close their estimates are from the original data, the lower bound can help data owners determine how much noise should be added to satisfy one given threshold of tolerated privacy breach.

Since the traditional matrix perturbation theory [6] mainly focuses on how the eigenvalues and the angle between eigenvectors (or invariance subspaces) of a perturbed matrix A is upper bounded by one perturbation, we cannot borrow any result to derive the lower bound. In this paper, we propose one Singular Value Decomposition (SVD) based reconstruction method and derive a lower bound for the reconstruction error. We then prove the equivalence between the Spectral Filtering based approach and the proposed SVD approach and as a result the achieved lower bound of SVD approach can also be considered as the lower bound of the Spectral Filtering based approach.

2 Spectral Filtering Reconstruction Method

We use the tilde conventions to denote perturbations. A symbol with a tilde over it denotes a perturbed quantity. The unperturbed quantity is denoted by the same symbol without a tilde. We denote $A = U^T U$ as the covariance matrix of U. The eigenvalues of A, $\Lambda(A) = \{\lambda_1, \cdots, \lambda_n\}$, and their corresponding eigenvectors $[e_1, \cdots, e_n]$ can be obtained, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. And those of $\tilde{A}$ are $\Lambda(\tilde{A}) = \{\tilde{\lambda}_1, \cdots, \tilde{\lambda}_n\}$.

We cast many of our analysis in terms of absolute and relative errors of matrix norm (i.e., $\|A\|_F$ and $\|A\|_2$ denote the Frobenius norm and 2-norm respectively), instead of component-wise bounds. The use of absolute and relative errors gives perturbation bounds a simplicity that makes them easier to interpret. Basically, the Frobenius form is used to measure the magnitude of data in total while the 2-norm is used to denote the largest eigenvalue of covariance matrix.

2.1 Spectral Filtering Revisited

The authors, in [5], provided an explicit filtering procedure as shown below.

1. Calculate the covariance matrix of $\tilde{U}$ by $\tilde{A} = \tilde{U}^T \tilde{U}$.
2. Apply spectral decomposition on $\tilde{A}$ to get $\tilde{A} = \tilde{Q}\tilde{\Lambda}\tilde{Q}^T$, where $\tilde{Q}$ is orthogonal matrix whose column vectors are eigenvectors of $\tilde{A}$, and $\tilde{\Lambda}$ is the diagonal matrix with the corresponding eigenvalues on its diagonals.
3. Using random matrix theory, the pair of $\lambda_{V_{min}}$ and $\lambda_{V_{max}}$, which provide the theoretical bounds of the eigenvalues corresponding to the matrix $V^T V$, are obtained.
4. Extract the first k components of $\tilde{A}$ as the principal components by $k = max\{i | \tilde{\lambda}_i \geq \lambda_{V_{max}}\}$. $\tilde{e}_1, \tilde{e}_2, \cdots, \tilde{e}_k$ are the corresponding eigenvectors, which form an orthonormal basis of a subspace $\tilde{\chi}$. Let $\tilde{Q}_k = [\tilde{e}_1\, \tilde{e}_2 \cdots \tilde{e}_k]$. The orthogonal projection on to $\tilde{\chi}$ is $P_{\tilde{\chi}} = \tilde{Q}_k \tilde{Q}_k^T$
5. Obtain the estimated data set using $\hat{U} = \tilde{U} P_{\tilde{\chi}}$.

The noise matrix V considered in [5] and this paper is generated using i.i.d. Gaussian distribution with zero mean and known variance. This represents the scenario where the noise is completely independent with original data.

The authors, in [3], investigated the explicit relation between $\hat{U} - U$ and the noise and derived the upper bound of $\|\hat{U} - U\|_F$ in terms of $\|V\|_F$ shown as

$$\|\hat{U} - U\|_F \leq \|\tilde{U}\|_F \frac{2\|V\|_F^2}{(\tilde{\lambda}_k - ||E||_2) - \sqrt{2}\|V\|_F^2} + \sqrt{k/n}||V||_F \tag{1}$$

where $E = V^T V$ is the derived perturbation on covariance matrix $A = U^T U$.

2.2 Strategies of Determining k

The original Spectral Filtering algorithm applied the following strategy to determine the first k eigen components.

Strategy 1. $k = max\{i|\tilde{\lambda}_i \geq \lambda_{V_{max}}\}$. *When the data set is large,* $\lambda_{V_{max}} \approx \lambda_{V_{min}} \approx \lambda_V$. *It becomes:* $k = max\{i|\tilde{\lambda}_i \geq \lambda_V\}$

We would point out that the previous Strategy 1 applied in [5] in general would not give the optimal reconstruction. The reason is that it aims to include all significant eigen components (with $\lambda_i > 0$) in projection space for reconstruction. However, since the inclusion of one eigen component also brings some additional noise projected on that eigen vector, the benefit due to inclusion of one insignificant eigen component may be diminished by the side effect due to the additional noise projected on this eigen vector.

In this paper, we propose a new strategy (as shown in Strategy 2) which compares the benefit due to inclusion of one component with the loss due to the additional projected noise. We shall show that Strategy 2 is expected to give one approximate optimal reconstruction.

Strategy 2. *The estimated data using* $\hat{U} = \tilde{U}P_{\tilde{\chi}} = \tilde{U}\tilde{Q}_k\tilde{Q}_k^T$ *is approximate optimal when* $k = min\{i|\tilde{\lambda}_i < 2\lambda_V\} - 1$.

Proof. In the Spectral Filtering method, when we select the first k components, the error matrix can be expressed as

$$\begin{aligned} f(k) &= \hat{U} - U \\ &= (U+V)\tilde{Q}_k\tilde{Q}_k^T - U \\ &= (U+V)\tilde{Q}\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix}\tilde{Q}^T - U \\ &= V\tilde{Q}\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix}\tilde{Q}^T - U[\tilde{Q}I\tilde{Q}^T - \tilde{Q}\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix}\tilde{Q}^T] \\ &= V\tilde{Q}\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix}\tilde{Q}^T - U\tilde{Q}\begin{pmatrix} 0 & 0 \\ 0 & I_{n-k} \end{pmatrix}\tilde{Q}^T \end{aligned} \tag{2}$$

Similarly, when we select the first k+1 components, the error matrix becomes

$$f(k+1) = V\tilde{Q}\begin{pmatrix} I_{k+1} & 0 \\ 0 & 0 \end{pmatrix}\tilde{Q}^T - U\tilde{Q}\begin{pmatrix} 0 & 0 \\ 0 & I_{n-k-1} \end{pmatrix}\tilde{Q}^T$$

$$
\begin{aligned}
&= V[\tilde{Q}\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix}\tilde{Q}^T + \tilde{e}_{k+1}\tilde{e}_{k+1}^T] - U[\tilde{Q}\begin{pmatrix} 0 & 0 \\ 0 & I_{n-k} \end{pmatrix}\tilde{Q}^T - \tilde{e}_{k+1}\tilde{e}_{k+1}^T] \\
&= (V\tilde{Q}\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix}\tilde{Q}^T - U\tilde{Q}\begin{pmatrix} 0 & 0 \\ 0 & I_{n-k} \end{pmatrix}\tilde{Q}^T) + V\tilde{e}_{k+1}\tilde{e}_{k+1}^T + U\tilde{e}_{k+1}\tilde{e}_{k+1}^T \\
&= f(k) + V\tilde{e}_{k+1}\tilde{e}_{k+1}^T + U\tilde{e}_{k+1}\tilde{e}_{k+1}^T \qquad (3)
\end{aligned}
$$

The last two parts in Equation 3 are the projections of noise and data on the (k+1)th eigenvector. Assume $\tilde{e}_i \approx e_i$, the strength of the data projection can be approximated as

$$
\begin{aligned}
||U\tilde{e}_{k+1}\tilde{e}_{k+1}^T||_F^2 &\approx ||Ue_{k+1}e_{k+1}^T||_F^2 \\
&= Tr[(Ue_{k+1}e_{k+1}^T)^T(Ue_{k+1}e_{k+1}^T] \\
&= Tr(e_{k+1}e_{k+1}^T U^T U e_{k+1}e_{k+1}^T) \\
&= Tr[e_{k+1}e_{k+1}^T(\sum_{i=1}^{n}\lambda_i e_i e_i^T)e_{k+1}e_{k+1}^T] \\
&= Tr(\lambda_{k+1}e_{k+1}e_{k+1}^T) \\
&= \lambda_{k+1}
\end{aligned}
$$

For i.i.d noise, the effect of the projection on any vector should be the same. Thus,

$$||V\tilde{e}_{k+1}\tilde{e}_{k+1}^T||_F^2 \approx \lambda_V$$

Hence, we include the i-th component only when $\lambda_i \geq \lambda_V$. The benefit due to inclusion of the i-th eigen component is larger than the loss due to the noise projected along the i-th eigen component.

It is easy to derive

$$Cov(U_i + V_i, U_j + V_j) = Cov(U_i, U_j) + E(V_i * V_j)$$

$$Var(U_i + V_i) = Var(U_i) + Var(V_i)$$

when the noise is independent to the data and also has no correlation among the noise.

Since $\tilde{\lambda}_i = \lambda_i + \lambda_V \geq 2\lambda_V$, hence

$$k = min\{i|\tilde{\lambda}_i < 2\lambda_V\} - 1$$

3 Our SVD-Based Reconstruction

In this section, we first present one SVD based reconstruction method and then derive the lower bound using the well-known Mirsky Theorem for SVD decomposition. We shall prove the equivalence between the proposed SVD method and Spectral Filtering method in Section 4. Hence the derived lower bound from SVD method can also be considered as the lower bound of Spectral Filtering method.

3.1 SVD Reconstruction

Singular Value Decomposition (SVD) decomposes a matrix $U \in R^{m\times n}$ (say $m \geq n$) into the product of two unitary matrices, $L \in R^{m\times m}$,$R \in R^{n\times n}$, and a pseudo-diagonal matrix $D = diag(\sigma_1, \cdots, \sigma_\rho) \in R^{m\times n}$, such that $U = LDR^T$ or $U = \sum_{i=1}^{n} \sigma_i l_i r_i^T$. The diagonal elements σ_i of D are referred to as *singular values*, which are, by convention, sorted in descending order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. The columns l_i and r_i of L and R are respectively called the left and right *singular vectors* of U. Similarly let $\tilde{U} = U + V$ be a perturbation of U and let $\tilde{U} = \tilde{L}\tilde{D}\tilde{R}^T$ be a perturbation of $\tilde{U}$.

---

input $\tilde{U}$, a given perturbed data set
V, a noise data set
output $\hat{U}$, a reconstructed data

BEGIN
1 Apply SVD on $\tilde{U}$ to get $\tilde{U} = \tilde{L}\tilde{D}\tilde{R}^T$
2 Apply SVD on V and assume σ_V is the largest singular value of V
3 Determine the first k components of $\tilde{U}$ by $k = min\{i|\tilde{\sigma}_i < \sqrt{2}\sigma_V\} - 1$
Assume $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \tilde{\sigma}_k$ and $\tilde{l}_i$, $\tilde{r}_i$ are the corresponding left and right singular vectors
4 Reconstructing U approximately as

$$\hat{U} = \tilde{U}_k = \sum_{i=1}^{k} \tilde{\sigma}_i \tilde{l}_i \tilde{r}_i^T$$

END

Fig. 1. SVD Based Reconstruction Algorithm

Figure 1 shows our SVD based reconstruction method. Please note that the strategy used for the SVD based reconstruction is $k = min\{i|\tilde{\sigma}_i < \sqrt{2}\sigma_V\} - 1$. We shall show its equivalence with the Strategy 2 of the Spectral Filtering method in Section 4.

3.2 Lower Bound Determination

Lower Bound. Consider $\hat{U} = \tilde{U}_k = \tilde{L}_k\tilde{D}_k\tilde{R}_k^T$ as the estimation of the original data set U. The estimation error between $\hat{U}$ and U has its lower bound:

$$||\hat{U} - U||_F \geq ||U_k - U||_F$$

where $k = min\{i|\tilde{\sigma}_i < \sqrt{2}\sigma_V\} - 1$.

Proof. U_k and U are matrices of the same dimensions with singular values

$$\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_n$$
$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$$

Since $\tilde{U}_k = \tilde{L}_k \tilde{D}_k \tilde{R}_k^T$

$$\tilde{\sigma}_{k+1} = \cdots = \tilde{\sigma}_n = 0$$

By Mirsky's theorem [6]

$$||\hat{U} - U||_F^2 \geq \sum_{i=1}^{n} |\tilde{\sigma}_i - \sigma_i|^2 \geq \sigma_{k+1}^2 + \cdots + \sigma_n^2 = ||U_k - U||_F^2$$

The relationship between the reconstruction bias and perturbation (especially the lower bound) will, in turn, guide us to add noise into the original data set. The lower bound gives data owners the worst case security assurance since it is bounded any matrix B of rank not greater than k derived by attackers. In order to preserve privacy, data owners need to make sure $||\hat{U} - U||_F / ||U||_F$ is greater than the privacy threshold τ, specified by users.

Based on the derived lower bound,

$$\tau ||U||_F \leq ||U_k - U||_F = \sigma_{k+1}^2 + \cdots \sigma_n^2$$

Hence k which might be chosen by attackers can be determined by

$$k = max\{i | \tau \leq (\sigma_{i+1}^2 + \cdots \sigma_n^2)/||U||_F\} \tag{4}$$

Based on our approximate optimal strategy,$\lambda_i \geq \lambda_V$, the data owner should add an i.i.d. noise V and let the eigenvalue of $(V^T V)$ satisfy

$$\lambda_{k+1} < \lambda_V \leq \lambda_k \tag{5}$$

Since λ_V is the eigenvalue of $V^T V$, the variance of the noise can be derived $Var(V) = \lambda_V/(m-1)$, where m is the number of row in V.

4 Equivalence of SVD and SF Reconstruction

SVD explicitly constructs orthonormal bases for the nullspace and range of a matrix. $U = LDR^T$. The non-zero singular values for U are precisely the square roots of the non-zero eigenvalues of the positive semi-definite matrix UU^T, and these are precisely the square roots of the non-zero eigenvalues of $U^T U$. Furthermore, the columns of L are eigenvectors of UU^T and the columns of R are eigenvectors of $U^T U$.

Theorem 1. *The reconstructed data from Spectral Filtering is*

$$\hat{U}_{SF} = \tilde{U} P_{\tilde{\chi}} = \tilde{U} \tilde{Q}_k \tilde{Q}_k^T$$

where $k = min\{i | \tilde{\lambda}_i < 2\lambda_V\} - 1$ *while the reconstructed data from SVD is*

$$\hat{U}_{SVD} = \tilde{L}_k \tilde{D}_k \tilde{R}_k^T$$

where $k = min\{i | \tilde{\sigma}_i < \sqrt{2}\sigma_V\} - 1$. *We have* $\hat{U}_{SF} = \hat{U}_{SVD}$ *and the k determined by* $k = min\{i | \tilde{\lambda}_i < 2\lambda_V\} - 1$ *and determined by* $k = min\{i | \tilde{\sigma}_i < \sqrt{2}\sigma_V\} - 1$ *are the same.*

Proof. We first prove these two methods are equivalent. Since $\tilde{R}_k = R \begin{pmatrix} I_k \\ 0 \end{pmatrix}$,

$$\tilde{U}\tilde{R}_k = \tilde{U}\tilde{R} \begin{pmatrix} I_k \\ 0 \end{pmatrix} = (\tilde{L}\tilde{D}\tilde{R}^T)\tilde{R} \begin{pmatrix} I_k \\ 0 \end{pmatrix} = \tilde{L}\tilde{D} \begin{pmatrix} I_k \\ 0 \end{pmatrix} = \tilde{L}_k\tilde{D}_k$$

Since the columns of right singular vectors ($\tilde{R}$) are the eigenvectors of $\tilde{U}^T\tilde{U}$, that is $\tilde{Q} = \tilde{R}$. Then

$$\hat{U}_{SF} = \tilde{U}\tilde{R}_k\tilde{R}_k^T = \tilde{L}_k\tilde{D}_k\tilde{R}_k^T = \hat{U}_{SVD}$$

We then prove the equivalence of determining k. Based on the fact that the singular value of U are the square root of eigenvalues of U^TU or UU^T, we have:

$$\tilde{\sigma}_i = \sqrt{\lambda_i(\tilde{U}^T\tilde{U})} = \sqrt{\tilde{\lambda}_i}$$
$$\sqrt{2}\sigma_V = \sqrt{2\lambda(V^TV)} = \sqrt{2\lambda_V}$$

So, $\tilde{\sigma}_i < \sqrt{2}\sigma_V \iff \tilde{\lambda}_i < 2\lambda_V$. Hence, $min\{i|\tilde{\sigma}_i < \sqrt{2}\sigma_V\} - 1 = min\{i|\tilde{\lambda}_i < 2\lambda_V\} - 1$.

5 Empirical Evaluations

In our experiment, we use the artificial dataset, as specified similarly in [5]. Specifically, U is a highly correlated data set with 35 variables. Each feature has a specific trend like sinusoidal, square, and triangular shape and there is no dependency between any two features. The additive noise follows i.i.d Gaussian distribution N(0,COV), where covariance matrix $COV = diag(\sigma^2, \cdots, \sigma^2)$ (The same as in [5]). From the previous discussion, we have $||V||_F \approx \sqrt{\sigma^2 mn}$. In our following experiments, we perturb the original data by different level of noises, which are generated by varying the covariance matrix COV. For each perturbed data, we use our SVD technique to reconstruct the point-wise data.

It is easy to see that different k leads to different reconstruction errors (which is measured by $re(U, \hat{U}) = ||U - \hat{U}||_F / ||U||_F$). From Table 1, we can see our SVD method (or the Spectral Filtering method with our Strategy 2) can achieve optimal results for all perturbations. Note the values with * denote the results achieved by our algorithm while the values in with † denote the results following the previous Strategy 1. The values in bold font highlight the best results achieved by varying k. Our proposed SVD method (or the Spectral Filtering method with our Strategy 2) can match the best results while the Spectral Filtering with the previous Strategy 1 suffers when relative large perturbations are introduced.

When we examine the original data, there exist 6 principle components as the data is highly correlated among 35 features. Hence, for relative small perturbations, the effects on the remaining 29 components are safely filtered. However, when we increase the noise level (i.e., $\|V\|_F$ increases), the noise will tend to affect the determination of k. This is because the gain of correct inclusion of some (not very significant) principal component is diminished by the loss due to the inclusion of noise projected on that component.

Table 1. The relative error $re(U, \hat{U})$ vs. varying V. The values with $*$ denote the results following Strategy 2, while the values with † denote the results following the Strategy 1. The bold values indicate those best estimations achieved by the spectral filtering technique. The noise is i.i.d. Gaussian noise with zero mean and equal variance on each dimensions.

noise		V1	V2	V3	V4	V5	V6	V7	V8	V9
variance		0.213	0.333	0.491	0.750	1.007	1.524	2.040	2.430	4.814
$\|\|V\|\|_F/\|\|U\|\|_F$		0.628	0.786	0.954	1.178	1.366	1.677	1.944	2.121	2.985
$re(U, \hat{U})$	k=1	0.821	0.825	0.830	0.839	0.847	0.863	0.877	0.890	0.960
	k=2	0.649	0.659	0.671	0.692	0.711	0.750	0.783	0.810	***0.956**
	k=3	0.440	0.461	0.488	0.529	0.565	0.636	0.694	***0.739**	0.964
	k=4	0.297	0.337	***0.383**	***0.450**	***0.506**	***0.607**	***0.687**	0.748	1.032
	k=5	0.271	***0.324**	0.383	0.465	0.532	0.651	0.745	0.816	1.141
	k=6	*†**0.260**	†0.325	†0.395	†0.489	†0.567	†0.699	†0.805	†0.883	†1.245
	k=7	0.282	0.353	0.428	0.530	0.614	0.757	0.873	0.956	1.348

6 Conclusion

Spectral filtering based technique has recently been investigated as a major means of point-wise data reconstruction [5,4]. It was empirically shown that under certain conditions this technique may be exploited by attackers to breach the privacy protection offered by randomization based privacy preserving data mining methods. We present an explicit lower bound of reconstruction accuracy in terms of Frobenius norm. This lower bound can help users determine how much and what kind of noise should be added when one tolerated privacy breach threshold is given. In the future we are interested in deriving the bounds at point-wise level.

References

1. D. Agrawal and C. Agrawal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th Symposium on Principles of Database Systems*, 2001.
2. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 439–450. Dallas, Texas, May 2000.
3. S. Guo and X. Wu. On the use of spectral filtering for privacy preserving data mining. In *Proceedings of the 21st ACM Symposium on Applied Computing*, pages 622–626, April 2006.
4. Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. Baltimore, MA, 2005.
5. H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 99–106, 2003.
6. G. Stewart and J. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

Frequent Pattern Discovery Without Binarization: Mining Attribute Profiles

Attila Gyenesei[1], Ralph Schlapbach[2], Etzard Stolte[1], and Ulrich Wagner[2]

[1] Knowledge and Data Analysis, Unilever Food and Health Research Institute, 3130 AC Vlaardingen, The Netherlands
{attila.gyenesei, etzard.stolte}@unilever.com
[2] Functional Genomics Center Zürich, Uni ETH Zürich, CH-8057 Zürich, Switzerland
{ralph.schlapbach, ulrich.wagner}@fgcz.ethz.ch

Abstract. Frequent pattern discovery has become a popular solution to many scientific and industrial problems in a range of different datasets. Traditional algorithms, developed for binary (or Boolean) attributes, can be applied to such data with a prerequisite of transforming non-binary (continuous or categorical) attribute domains into binary ones. As a consequence of this binarization, the discovered patterns no longer reflect the associations between attributes but the relations between their binned independent values, and thus, interactions between the original attributes may be lost. In this paper we propose to overcome this limitation by introducing the concept of mining frequent *attribute profiles* that describes the relationships between the original attributes. By this concept, previously hidden interactions can be discovered and redundant patterns that are identified by traditional methods are eliminated. A novel algorithm, called MAP, has been developed for mining attribute profiles that can be potentially applied to diverse data domains. The effectiveness of the proposed method is shown by using gene expression or microarray data.

1 Introduction

The problem of association pattern discovery (APD) originates from market basket analysis which aims at finding interesting relationships hidden in large datasets. Such relationships can be represented in the form of *frequent itemsets* and *association rules*. APD is a two-step process: the first and most time-consuming step is to find frequent sets of items (called itemsets) that occur together in at least as many transactions as a given support threshold. The *support* of an itemset is the number of transactions that contain the itemset. The number of potential frequent itemsets is exponential to the number of items, which presents the main problem of the first step. The second step generates association rules from the frequent itemsets. Based on the mining strategy by which frequent itemsets are discovered, two types of algorithm can be distinguished: breadth-first search and depth-first search. The most commonly used breadth-first search algorithm is Apriori [2,14] and its variants [4,8]; whereas Eclat [16] and FP-growth [9] are popular depth-first search algorithms. For other APD algorithms together with precise descriptions and analyses, see [10].

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 528 – 535, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Since its introduction, APD has been successfully applied not only to market basket analysis but to many other scientific and industrial problems, and more recently to gene expression data [5,7,12]. In market basket analysis an item is purchased or not purchased in a transaction, which requires the data to be represented by binary attributes. Real-world datasets, however, often contain continuous and categorical values. In gene expression data, for instance, a real value is assigned to each gene that specifies its expression level in a given tissue or condition. Applying conventional pattern mining algorithms to such datasets requires a preliminary transformation of non-binary attributes to binary ones [15], which can only partly discover association patterns. This limitation of traditional methods can be highlighted by a simple example taken from a gene expression experiment as shown in Table 1. The expression values of gene *A*, gene *B*, gene *C* and gene *D* were determined for four different conditions as significantly repressed (1), significantly expressed (2), or neither significantly repressed nor expressed (3).

Table 1. Sample gene expression data

	gene *A*	gene *B*	gene *C*	gene *D*
cond1	2	3	2	3
cond2	1	2	1	2
cond3	1	1	1	2
cond4	2	1	2	3

The research problem in such data is to find relationships between co-regulated genes, or in other words, to discover frequent combinations and associations of genes that display co-occuring changes condition by condition. Using the traditional methods, first the sample data is transformed into binary data before any of the frequent pattern mining algorithms can be applied. Table 2 summarizes the frequent patterns and their supports for the binned data.

Table 2. Frequent patterns vs. frequent profiles

support	Frequent patterns	Frequent profiles
4	-	$A\{0\}C\{1\}D$ and its 3 subsets
3	-	-
2	{*A*:1, *C*:1, *D*:2}, {*A*:2, *C*:2, *D*:3} and their 6 subsets	$A\{1\}B\{-1\}C\{1\}D$ and its 8 subsets

The highest co-occurrence that can be identified by traditional methods is 50% and it is satisfied by two frequent patterns: {*A*:1, *C*:1, *D*:2} for condition 2 and 3, and {*A*:2, *C*:2, *D*:3} for condition 1 and 4. However, the data in Table 1 show that there is a real association between genes *A*, *C* and *D* for all conditions. Where the expression level of one of the genes is affected in a particular way (repressed, expressed or no change) the expression values of the other two genes are affected in the same way in all conditions. This allows for the identification of genes whose expression profiles

follow the same patterns in response to different conditions. Intriguingly, traditional techniques are unable to identify the relationship even between genes *A* and *C* despite the fact that they have identical expression values in all conditions because of the low support of their binned values. Moreover, with support threshold 2, an accurate method should result in a single pattern for genes *A*, *C* and *D* thus reducing the number of "redundant" patterns (patterns containing the same genes with different binned values).

In this paper we tackle these problems by introducing the concept of mining frequent attribute profiles. Attribute profiles consider the original attributes without the need for binarization, and present their "trends" that are not visible when only binary values are used. By this concept, associations between the original attributes can be discovered that remain hidden to traditional approaches.

Since traditional frequent itemset mining algorithms cannot be applied to discover the introduced attribute profiles, in this paper we propose an efficient depth-first search attribute profile mining algorithm. More precisely, the original data set is first compressed into an attribute distance tree structure where information about the trends (distances) of attributes is stored. Secondly, a recursive searching technique identifies and collects all of the frequent attribute profiles from the attribute distance tree. The effectiveness of the proposed method is demonstrated for a real-world, gene expression dataset in Section 4.

2 Frequent Attribute Profiles

Our starting point is that continuous attributes are only discretized and the data has as many fields as the number of attributes.

Let X and Y be two attributes and let d denote the difference between two attribute values of X and Y in a transaction t such that $d = t[Y] - t[X]$. The formula $X\{d\}Y$ is called an *attribute profile* between attributes X and Y. It is also called *attribute profile of length* 2 since it contains two attributes. The *support* of an attribute profile $p \subseteq P$ in dataset T is the number of transaction that contains the profile in T:

$$supp(p) = |\{\ t \mid p \subseteq t, t \in T\}|$$

The *frequency* of an attribute profile p in T is the probability of p occurring in a transaction $t \in T$:

$$freq(p) = supp(p) / |T|$$

An attribute profile p is called *frequent* if its support (frequency) is greater than or equal to a (user defined) minimum support (frequency) threshold σ_{supp} (σ_{freq}):

$$supp(p) \geq \sigma_{supp} \quad (freq(p) \geq \sigma_{freq})$$

The research problem of attribute profile mining is to find all attribute profiles with sufficient support (frequency). Table 2 summarizes the frequent attribute profiles and their supports for the data given in Table 1.

In contrast to the frequent itemset mining which is unable to identify the association between genes *A*, *C* and *D* in the example from Section 1, attribute profile mining can discover this relation, even when a high (100%) frequency threshold is set. Moreover, we can easily see the trends (distances) of attributes from the frequent attribute

profile. Note that each frequent pattern discovered by traditional methods can be obtained from frequent attribute profiles. For example, the two frequent patterns {*A*:1, *C*:1, *D*:2} and {*A*:2, *C*:2, *D*:3} with support values 2 are included in the attribute profile $A\{0\}C\{1\}D$ with support value 4.

To summarize the advantages of attribute profile mining over frequent itemsets mining, we can conclude that

1. Attribute profiles describe relationships between the entire attributes in contrast to traditional patterns, which identify associations between independent binned attribute values.
2. By the definition of attribute profiles, trends or distances of non-binned attributes are taken into account in order to identify associations between the entire attributes having non-frequent binned values but frequent trends transaction by transaction. Thus, previously hidden (lost) patterns of related attributes can be discovered, which are not found among the patterns produced by traditional APD methods.
3. By discovering attribute profiles, a considerable number of redundant associations can be eliminated. By using traditional methods, redundant associations can appear, when the binned expression values of a set of particular genes have sufficient frequency in more than one condition (or sample).

Note that the distance between two attributes, introduced in this section, can be defined in different ways based on the data domains. Therefore, the distance can be symmetric or asymmetric. Applying symmetric distance measure, patterns of negatively (inversely) correlated genes can be also discovered by attribute profiles that are hidden by traditional patterns. Also note that our distance measure is close to the semantics of functional (multivalued) dependencies, see [1, 13].

3 Mining Attribute Profiles

In this section we present our Mining Attribute Profile algorithm (*MAP*). The proposed algorithm can be characterized as a depth first search, divide-and-conquer algorithm, such as FP-growth [9]. We chose this type of searching strategy in order to reduce the number of database scans and avoid the costly set-containment-test operation that can be the case in applying the breadth-first search strategy, such as Apriori [2].

The mining is carried out in two steps in which the first step constructs a compact data structure called a Frequent Attribute Distance Tree (or FAD-tree), and the second step extracts the frequent attribute profiles directly from this FAD-tree structure.

3.1 Constructing a Frequent Attribute Distance Tree

In order to construct a compact data structure for efficient attribute profile mining, we apply the idea of building a frequent pattern tree (FP-tree) [9]. Similar to the FP-tree, the FAD-tree is constructed by reading the database transaction by transaction and mapping each transaction onto a path in the FAD-tree. A path compression occurs when two or more transactions have the same attribute profiles starting from the first attribute. The main difference between our FAD-tree and the original FP-tree is that

while the FAD-tree is built by the entire attributes (each node is a frequent attribute), the FP-tree is constructed by binned attribute values as single items. Moreover, the FAD-tree considers and stores attribute distances (or trends) between two successive attributes.

3.2 Mining Frequent Attribute Profiles Using the FAD-Tree

The MAP algorithm generates frequent attribute profiles from the constructed FAD-tree by exploring the tree in a top-down and recursive manner. It splits the problem into sub-problems by decomposing the FAD-tree into disjoint sub-FAD-trees and the header table into sub-header tables. This decomposition is carried out attribute by attribute in a stepwise manner. For each attribute node, a parent distance is calculated between the node and the root of the FAD-tree by summing the distance values of intervening nodes. Nodes with calculated equivalent distances are grouped together. For each group with a sufficient support value, a sub-FAD-tree is constructed, i.e. rooted by that attribute. A corresponding sub-header table is also constructed in which the child of the rooted attribute forms the first position. Note that the nodes in the FAD-tree are gathered by using the linked lists in the header table. During the decomposition process, paths with the same parent distance are merged.

The above procedure is applied in a recursive way so that each sub-FAD-tree is used as a FAD-tree in the next recursion. During the decomposition, the roots of sub-FAD-trees are stored as frequent profiles. If the constructed sub-FAD-tree has only a single branch, then there is no need to build a new sub-FAD-tree, all frequent attribute profiles can be enumerated directly from the single branch.

A high-level pseudo-code of the MAP algorithm is given in the following:

Algorithm MAP – Mining Attribute Profiles

Input: FAD-tree *T*, header table *H*, support threshold σ_{supp}
Output: The complete set of frequent attribute profiles
Description:

```
// Check whether T has only a single path.
if T has only a single path then
  enumerate frequent attribute profiles from FAD-tree T
else
  // Main loop for each attribute in the header table
  // with sufficient support value.
  for all attribute a ∈ H, a.supp ≥ σ_supp do
     // Calculate parent distance between the nodes and the
     // root.
     for all n ∈ a.nodes do
        calculate n.parent_dist (the distance between n and
        the T.root)
     Group nodes with same equivalent parent distances
     // For each group with sufficient support, create
     // sub-FAD-tree and sub-header table.
     for each set of a.nodes with same parent_dist and
     sufficient support do
        create sub_FAD-tree sub_T and sub_header table sub_H
        update set of frequent attr. profiles by sub_T.root
        call MAP(sub_T, sub_H, σ_supp)
```

A sub-FAD-tree rooted by attribute *A* is used to demonstrate how the MAP algorithm works for the sample data (Table 1), where the support threshold is set to 2 (Figure 1).

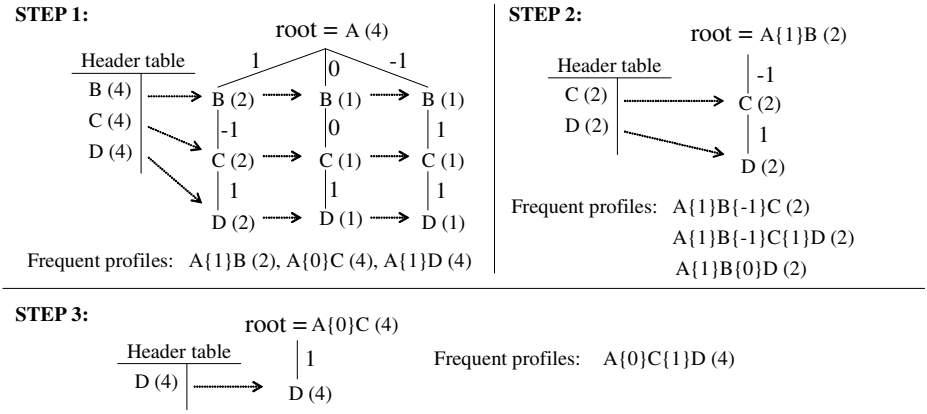


Fig. 1. Applying the MAP algorithm to the FAD-tree rooted by attribute *A*

4 Experimental Results

In this section, we demonstrate the usability and efficiency of our proposed attribute profile mining method (MAP) by comparing the performance of MAP and the traditional frequent itemset mining method (FIM) when applied to the gene expression data set of Hughes et al. [11]. This data contains information about the expression levels of 6316 genes throughout 300 diverse yeast mutants or wild type yeast challenged with different chemical treatments.

We discretized each expression value of the normalized data to 3 integers 1, -1 and 0 representing expressed, repressed and neither expressed nor repressed. This was achieved by assigning all expression values greater than 0.2 for the log base 10 of the fold change to a value of 1 (expressed), all values less than -0.2 to a value of -1 (repressed) and those between -0.2 and 0.2 to 0 (neither expressed nor repressed). To limit the effect of noise, 0 is considered to be a missing value. For FIM, we needed an additional binarization step to bin the gene attributes of discretized data into single Boolean items.

Both FIM and MAP methods were tested on 7 different support thresholds. As an example, here we analyze the lengths of the longest patterns (frequent itemsets and profiles with the greatest number of genes) and the number of maximal frequent patterns. The reason for using the maximal patterns is to consider a hidden association only once, i.e. the longest one (as all subsets of a frequent pattern are frequent as well). Maximal frequent patterns are useful to identify a small representative set of patterns from which all other frequent patterns can be derived. A frequent pattern is called maximal if it has no superset that is frequent [3]. For example, in our sample

dataset, only the listed frequent itemsets and profiles are maximal (Table 2), i.e. having more than one gene. Table 3 summarizes the results for both methods. The frequency threshold values, σ_{rel}, were selected to keep the number of maximal frequent patterns manageable.

Table 3. Comparison of FIM and MAP methods

σ_{rel}	#genes in the longest patterns		# max. patterns (itemsets/profiles)		#max.hidden associations	Running times (in seconds)	
	FIM	MAP	FIM	MAP		FIM	MAP
0.19	4	5	15	163	148	0.16	0.05
0.18	4	6	27	303	276	0.2	0.11
0.17	5	6	48	585	537	0.21	0.12
0.16	6	8	96	1162	1066	0.25	0.19
0.15	7	10	195	2330	2135	0.37	0.35
0.14	8	13	425	4462	4037	1.55	3.45
0.13	11	15	871	7374	6503	3.25	12.56

For all support thresholds, MAP gives the longest and most associations. For example, at a minimum frequency of 0.14 (equal to 42 treatments), the number of genes in the longest associations identified by FIM was 8 whilst MAP identified frequent associations between 13 genes. As a consequence, MAP discovered more maximal frequent patterns than FIM for the same threshold, where all of the associations recognized by FIM were also identified by MAP. The last column shows the number of maximal hidden associations, i.e. the associations that were identified by MAP and not by FIM. The result clearly shows that previously hidden associations can be discovered by our introduced frequent profile mining method.

To compare the running times of traditional FIM and the implemented MAP methods, we chose the FP-growth implementation, provided by Bart Goethals [6]. In situations when higher thresholds are set, MAP is the fastest, whereas in other cases it has the longest running time. However, this is due to the fact that it discovers much more co-regulations between genes for the same thresholds than are found by FIM methods. This is probably a general observation: more frequent profiles and thus, more candidate associations result in slower speed. Similar numbers of profiles and itemsets generate comparable running times. We also wish to point out that our implementation is an initial version with no additional enhancements to increase calculation speed.

5 Conclusions

In this paper, a novel attribute profile mining method was introduced for frequent pattern discovery, as an improvement to the itemset mining approaches common today. The main idea of the method is that non-binary attributes are mined without a preliminarily binarization. As a consequence, frequent patterns of entire attributes hidden to traditional methods can be discovered. An algorithm was developed for the proposed problem and shown to perform effectively when applied to gene expression

data. We expect that the method could also be effectively applied to other large scale data in the area of systems biology, such as protein quantification data, single nucleotide polymorphism data, and data from promoter studies.

References

[1] Agier, M., Petit, J-M., Suzuki, E.: Towards Ad-Hoc Rule Semantics for Gene Expression Data. ISMIS (2005) 494 – 503

[2] Agrawal, R., Srikant, R.: Fast Algorithm for Mining Association Rules in Large Databases. Proceedings of the 20th International Conference on Very Large Data Bases (1994) 487 – 499

[3] Bayardo, R.J.: Efficiently Mining Long Patterns from Databases. Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, (1998) 85 – 93

[4] Brin, S., Motwani, R., Ullman, J., Tsur, S.: Dynamic Itemset Counting and Implication Rules for Market Basket Data. Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, (1997) 255 – 264

[5] Creighton, C., Hanash, S.: Mining Gene Expression Databases for Association Rules. Bioinformatics, Vol. 19 (1), (2003) 79 – 86

[6] Frequent Itemset Mining Implementations Repository website, http://fimi.cs.helsinki.fi

[7] Georgii, E., Richter, L., Ruckert, U., Kramer, S.: Analyzing Microarray Data Using Quantitative Association Rules. Bioinformatics, Vol. 21 (2), (2005) 123 – 129

[8] Gyenesei, A., Teuhola, J.: Probabilistic Iterative Expansion of Candidates in Mining Frequent Itemsets. Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, (2003) 413 – 422

[9] Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns without Candidate Generation. Proceedings of the 2000 ACM SIGMOD Int. Conf. on Management of Data, (2000) 1 – 12

[10] Hipp, J., Güntzer, U., Nakhaeizadeh, G.: Algorithms for Association Rule Mining – A General Survey and Comparison. SIGKDD Explorations, Vol. 2(1), (2000) 58 – 64

[11] Hughes, T.R., Marton, M.J., Jones, A.R., Roberts, C.J., et al.: Functional Discovery via a Compendium of Expression Profiles. Cell, Vol. 102, (2000) 109 – 126

[12] Ji, L., Tan, K.L.: Mining Gene Expression Data for Positive and Negative Co-Regulated Gene Clusters. Bioinformatics. Vol. 20 (16), (2004) 2711 – 2718

[13] Mannila, H., Toivonen, H.: Levelwise Search and Borders of Theories in Knowledge Discovery. DMKD 1, (1997) 241 – 258

[14] Mannila, H., Toivonen, H., Verkamo, A.I.: Efficient Algorithms for Discovering Association Rules. Proceedings of the 1994 AAAI Workshop on Knowledge Discovery in Databases, (1994) 181 – 192

[15] Srikant, R., Agrawal, R.: Mining Quantitative Association Rules in Large Relational Tables. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, (1996) 1 – 12

[16] Zaki, M.J.: Scalable Algorithms for Association Mining. IEEE Transactions on Knowledge and Data Engineering, Vol. 12 (3), (2000) 372 – 390

Efficient Name Disambiguation for Large-Scale Databases

Jian Huang[1], Seyda Ertekin[2], and C. Lee Giles[1,2]

[1] College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA 16802, U.S.A.
{jhuang, giles}@ist.psu.edu
[2] Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802, U.S.A.
sertekin@cse.psu.edu

Abstract. Name disambiguation can occur when one is seeking a list of publications of an author who has used different name variations and when there are multiple other authors with the same name. We present an efficient integrative framework for solving the name disambiguation problem: a blocking method retrieves candidate classes of authors with similar names and a clustering method, DBSCAN, clusters papers by author. The distance metric between papers used in DBSCAN is calculated by an online active selection support vector machine algorithm (LASVM), yielding a simpler model, lower test errors and faster prediction time than a standard SVM. We prove that by recasting transitivity as density reachability in DBSCAN, transitivity is guaranteed for core points. For evaluation, we manually annotated 3,355 papers yielding 490 authors and achieved 90.6% pairwise-F1. For scalability, authors in the entire CiteSeer dataset, over 700,000 papers, were readily disambiguated.

1 Introduction

Name disambiguation is desired in many cases: e.g., evaluating faculty publications, calculating statistics of social network and author impacts, etc. The metadata of publications such as authors, titles etc. is very valuable for automatic bibliometrics and citation analysis. Manual extraction of metadata can be costly for large-scale digital libraries such as the Google Scholar and CiteSeer. Automatic metadata extraction [1] is not perfect especially for papers crawled from the web, where many items are missing or incomplete. With author profiles constructed from disambiguation, these fields can be correctly populated, improving the quality of existing metadata.

Name disambiguation is an interesting data mining problem with AKA's and other pseudonyms. The problem is deemed challenging in large-scale digital libraries. First, name disambiguation is a *meta* problem. Unlike disambiguation in NLP, name disambiguation in academic papers does not necessarily have context in a document, since authors do not appear in the text. In our case we use the metadata of an author's papers to determine his identity. Moreover, *scalability* is a significant concern for large-scale databases, thus giving a preference

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 536–544, 2006.
© Springer-Verlag Berlin Heidelberg 2006

for unsupervised or semi-supervised methods since it's implausible to annotate and train a classifier for each namesake. In addition, *expandability* is an issue for persistent disambiguation. As new papers come in, more information is available to refine previous results and name clusters could be adjusted when appropriate.

Our **contribution** is addressing the above challenges as follows:

- We use an online SVM algorithm (LASVM) to build a supervised distance function, which yields a simpler yet faster model with active learning.
- We overcome the transitivity problem commonly found in other disambiguation work by using an efficient clustering algorithm DBSCAN.
- Our framework is easily expandable to new papers: the supervised learner for the distance function can easily handle additional data with online learning; also, DBSCAN can adjust name clusters based on the new information.
- The framework integrates supervised and unsupervised methods to provide a scalable solution, and is readily amendable to various improvements.

2 Related Work

Prior name disambiguation work mainly deals with the **citation matching** problem [2, 3, 4]. Hybrid Naive Bayes and Support Vector Machine [5] methods are inappropriate for large-scale databases, due to the cost of human annotation. K-spectral clustering was used in [4] to find an approximation of the global optimal solution. However, the computation complexity $O(N^2)$ is intractable for large-scale databases. Also, K is unknown *a priori* for an increasing database. The scalability issue is addressed in [6] by using a two-level blocking framework, reducing computation complexity to $O(C\,|B|)$ (C is the number of blocks and $|B|$ the average size of blocks). However, citations are differentiated by single pairwise distance without clustering. Like earlier agglomerative clustering approaches [7], this could lead to the **transitivity problem**, due to the noisy data and the inaccurate distance function. Multiple distances instead of a threshold on single pairs are accounted for in [8], imposing transitivity by adding an additional feature (with weight $-\infty$) into the Conditional Random Field model. We ameliorate the problem by using an efficient unsupervised clustering method DBSCAN [9], which also makes coreferent decisions based on multiple distances.

3 Methods

3.1 Solution Overview

We formalize *name disambiguation* in Fig.1 as:

> Given a research paper $p^{(i)}$, each author appearance $a_u^{(i)}$ in this paper is associated with a metadata record $r_u^{(i)}$, consisting of a set of attributes $\{t_{u,k}^{(i)}\}_{k=1}^{m}$. Our goal is to find an assignment function Θ, such that $\Theta(a_u^{(i)}) = E_w$, where E_w represents the real entity; in other words, $\Theta(a_u^{(i)}) = \Theta(a_v^{(j)})$ if and only if $a_u^{(i)}$ and $a_v^{(j)}$ refer to the same person.

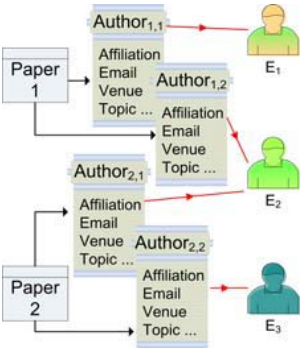


Fig. 1. Author disambiguation

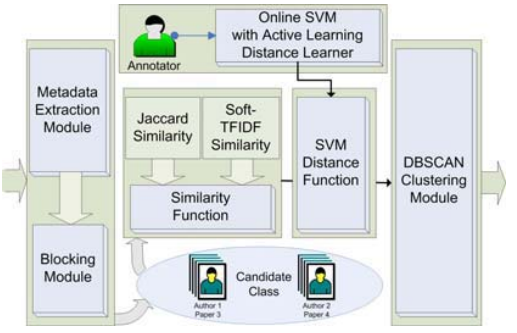


Fig. 2. Disambiguation system overview

Fig.2 shows the system architecture. The metadata extraction module [1] first extracts author metadata records from each paper. The blocking module then blocks namesakes into candidate classes including only non-conflicting name variations, thus significantly reduces the number of similarity calculation for pairs from the entire database to within candidate classes. Afterward, the similarity function computes a similarity vector $s^{(i,j)}=[sim_1(t^{(i)}_{u,1},t^{(j)}_{v,1}),...,sim_m(t^{(i)}_{u,m},t^{(j)}_{v,m})]^T$, from the attributes $\{t^{(i)}_{u,k}\}^m_{k=1}$ and $\{t^{(j)}_{v,k}\}^m_{k=1}$ in record pairs, corresponding to author appearances $a^{(i)}_u$ and $a^{(j)}_v$ in a candidate class. We use different similarity predicates sim_l depending on the nature of the attributes. For instance, the edit distance is used for emails and URLs; token-based Jaccard similarity for addresses and affiliations; hybrid similarity Soft-TFIDF [10] for name variations.

The SVM then uses the similarity vector $s^{(i,j)}$ as a feature vector to classify whether $r^{(i)}_u$ and $r^{(j)}_v$ are coreferent, and the confidence of coreference is used as a pairwise distance metric. Finally, DBSCAN constructs clusters based on multiple pairwise distances, which addresses the transitivity problem. These last two modules are described in more detail in the rest of this section.

3.2 Distance Function with Online SVM and Active Learning

In the hypothetical space $\mathcal{R}$ spanned by metadata records, we need to determine the distance $dist(s^{(i,j)})$ between two records $r^{(i)}_u$ and $r^{(j)}_v$. The distance function $dist$ is non-trivial and data-driven, thus we use a supervised learning algorithm to determine such a function. Support Vector Machine (SVM) [11] is originally designed for binary classification and shows good generalization performance. We, however, use the SVM's to obtain the learner's confidence in the coreferent class as in [10]. The confidence values determine the distances between the record pairs, i.e., the more confident the SVM model classifies two metadata records as coreferent, the closer they are in $\mathcal{R}$. For simplicity of notation, we refer to the training sample $s^{(i,j)}$ as $\mathbf{x_k}$ and its true label as y_k. Given a labeled training dataset $\{(\mathbf{x_1}, y_1), ..., (\mathbf{x_N}, y_N)\}$ $(y_i \in \{-1, +1\})$, the SVM aims to find

an 'optimal' hyperplane $(\mathbf{w} \cdot \mathbf{x}) + b = 0$ ($\mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}$) that separates the training data, after solving the optimization problem of minimizing the function $\mathbf{L}(\mathbf{w}) = \|\mathbf{w}\|^2 /2$, subject to $y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1$ ($i = 1, ..., N$).

We efficiently train the SVM model with an online kernel classifier LASVM [12]. LASVM relies on the traditional soft margin SVM formulation, while it works faster and preserves the classification accuracy rates of the state-of-the-art traditional SVM solvers. Traditional SVM works in a batch setting; whereas LASVM works in an online setting, where its model is continually modified as new training instances become available. The speed improvement and the less memory demand with **online learning** makes LASVM applicable to very large datasets. When the digital library is populated with new papers, LASVM can integrate the information of the new data without retraining all the samples, thus it is adaptable to growing datasets.

In our setting, the metadata records are inherently noisy, thus not all the training samples are equally informative. We believe that by using only the most informative samples and discarding the noisy samples, we will get a simpler and sparser model. This can be accomplished by **active sample selection**. In SVM based active learning, the most informative sample among all the training data is the one closest to the hyperplane. Classical active learning method with SVM's [13] is computationally expensive as it requires a search through all the unseen training samples. We use a method as in [12] that will not necessitate a full search, but locates an approximate most informative sample by examining a small constant number of randomly chosen samples. The method first picks M ($M = 50$ as in [12]) random training samples and selects the best one among them. Thus active sample selection can be done in reasonable time.

3.3 DBSCAN Clustering

We use the clustering method DBSCAN [9] to cluster author appearances in papers and prove that it can handle the inconsistency of author labeling. We do not simply classify whether two metadata records are coreferent or not simply based on the pairwise distance $dist(s^{(i,j)})$, due to the **transitivity problem**: for a triad (o, p, q), point o is coreferent with p, and p with q, while o is not coreferent with q. This is an inconsistent condition since coreference should be transitive, and is due to errors in metadata extraction, imperfect similarity metric and misclassification. We formally prove that DBSCAN for the most cases resolves the transitivity problem. Other reasons that we choose DBSCAN include:

- Minimal domain knowledge is needed to determine the two parameters ϵ and MinPts, which can be tuned by visualization methods such as OPTICS [14].
- DBSCAN can model clusters of any arbitrary shape and delimit clusters more intuitively to human interpretation.
- DBSCAN is highly efficient, for its computation complexity is $O(N \log N)$.

We briefly review the definitions in DBSCAN that are used in our theorems. For an author appearance $a_u^{(i)}$ in a candidate class $\mathcal{D}$, DBSCAN induces the cluster $\mathcal{C}$ from $\mathcal{D}$ such that $\forall a \in \mathcal{C}, \Theta(a) = E_w$.

Definition 1. *Point p is* ***directly density reachable*** *from q, if* $p \in N_\epsilon(q)$ *and* $|N_\epsilon(q)| \geq MinPts$*(core point condition), where* $N_\epsilon(q) = \{p \in \mathcal{D} \,|dist(p,q) \leq \epsilon\}$.

Definition 2. *Point p is* ***density reachable*** *from q if there exists a chain* $p_1 = p, ..., p_n = q$*, such that* p_{i+1} *is directly density reachable from* p_i.

Definition 3. *Point p is* ***density connected*** *to q if there exists o, such that both p and q are density reachable from o.*

Definition 4. *A cluster* $\mathcal{C}$ *is a subset of* $\mathcal{D}$ *satisfying:*
1.(Maximality) $\forall p, q$*, if* $p \in \mathcal{C}$ *and q is density reachable from p, then* $q \in \mathcal{C}$.
2.(Connectivity) $\forall p, q \in \mathcal{C}$*, p is density connected to q.*

Under the DBSCAN framework, we recast the **coreference** relationship as **density connectivity**, both of which are symmetric. Formally speaking, for all $p, q \in \mathcal{D}$, p is coreferent with q iff p is density connected to q. We prove the following theorem which shows that the transitivity problem with DBSCAN is no longer an issue for the most part.

Theorem 1. *Transitivity is guaranteed as long as p is a core point.*

Proof. A contradiction exists if the transitivity problem exists, i.e., o is not coreferent with q. o is coreferent with p implies that o is density connected to p, so there exists r such that p and o are density-reachable from r. Hence two chains $a_1 = r, ..., a_k = o$ and $b_1 = r, ..., b_l = p$ exist, such that a_{i+1} and b_{i+1} are directly density reachable from a_i and b_i respectively. Specifically, $p \in N_\epsilon(b_{l-1})$ implies $b_{l-1} \in N_\epsilon(p)$. Since p is a core point, we have $|N_\epsilon(p)| \geq MinPts$. Thus b_{l-1} is directly density reachable from p. Note that by definition of density reachable, $b_1 ... b_{l-1}$ should all satisfy core point condition. This forms a reverse chain $b_l = p, ..., b_1 = r$ such that b_{i-1} is directly density reachable from b_i. Now we have formed a density reachable chain from p to o ($b_l = p, ..., b_1 = a_1 = r, ..., a_k = o$), and similarly another chain from p to q. Thus o is density connected to q, which violates the assumption that o is not coreferent with q. □

Theorem 2 determines the correctness of DBSCAN for coreference resolution, and corollary 1 dictates the absence of transitivity problem within a cluster.

Theorem 2. $\forall \mathcal{C}$ *and* $\forall p, q \in \mathcal{C}$*, p is coreferent with q.*

Proof. By connectivity property in definition 4, $\forall p, q \in \mathcal{C}$, p is density connected to q. Therefore, p is coreferent with q.

Corollary 1. *The transitivity problem does not exist for any triad in a cluster.*

Combining Theorem 1 and Corollary 1, we are left with the case where the transitivity problem exists: p is a border point ($|N_\epsilon(q)| < MinPts$) of different clusters. The nature of density-based clustering implies that this is a rare case since such points will lie on the cluster boundary and will be sparse. Such points are due to insufficient information which would be necessary to disambiguate a

Table 1. Author datasets (R=#records, A=# authors)

ID	Dataset	R	A
1	A. Gupta	506	44
2	A. Kumar	143	36
3	C. Chen	536	103
4	D. Johnson	350	41
5	J. Anderson	327	43
6	J. Robinson	115	30
7	J. Smith	743	86
8	K. Tanaka	53	20
9	M. Jones	352	53
10	M. Miller	230	34
	Total	3,355	490

Table 2. SVM models testing results: LASVM vs. LIBSVM

ID	Error(%)		Prediction Time(sec.)[a]	
	LIBSVM	LASVM(%chg.)	LIBSVM	LASVM(%chg.)
1	19.34	**17.989**(-7.00%)	137.3	**109.3(-20.4%)**
2	6.491	**6.149**(-5.26%)	6.3	**5.1(-19.0%)**
3	**4.882**	4.885(+0.07%)	118.8	**94.2(-20.7%)**
6	2.814	**2.335**(-17.0%)	5.3	**4.1(-22.6%)**
7	9.721	**9.168**(-5.69%)	215.6	**170.2(-21.1%)**
8	11.00	**10.513**(-4.45%)	1.1	**0.8(-27.3%)**
10	21.31	**18.35(-13.9%)**	25.3	**19.6(-22.5%)**
Avg	11.218	**9.913(-7.60%)**	72.8	**57.6(-23.5%)**

[a] Test on Dell Precision 370 server (3.0GHz Xeon CPU)

particular person's name on a paper. When more information is available, the problem can be easily solved with DBSCAN by merging or splitting clusters.

To sum up, by using the SVM to learn the underlying distance function, DBSCAN acts as an assignment function Θ to disambiguate authors in papers.

4 Experiments

We empirically study the efficiency and effectiveness of our proposed method by testing both the supervised distance function and the entire framework. Using the CiteSeer metadata (obtained from SVM-based metadata extraction [1]), 10 most ambiguous names are sampled from the entire dataset as listed in Table 1. These names are in parallel with the names used in [5, 4] representing the worst case scenario, and are geographically diverse to cover names of different origins. 3,355 papers are manually labeled yielding 490 authors. For those ambiguous author names from different papers, we meticulously went through the original papers, homepages, CVs, etc, to confirm their authorship.

4.1 Experiments on SVM Based Distance Function

We select datasets with ID number 4, 5 and 9 as a three-fold training dataset, consisting of 81,073 pairwise coreference training samples. Our first goal is to obtain a simpler model for efficient distance calculation. As we see in Fig. 3, in active learning setups, after using certain number of training data, the number of support vectors saturates and the test error stabilizes. We observe that adding more training data after this point hardly changes the model. This implies that the most informative samples are already included in the model and the remaining samples do not provide extra information. Therefore, we determine an early stopping point for training by cross validation results (Fig. 3). We first select an interval of iteration number from 12,310 to 14,100, where the average cross validation error is stably minimized. Then we fix the iteration number to 14,100, where the number of support vectors is closest to saturation.

Our LASVM model is trained on the entire training dataset, stopping the training process at this iteration number. For comparison, we also train a classical SVM model with a popular implementation LIBSVM [15] using batch learning. Table 2 shows the test error and prediction time of LASVM, compared to classical SVM, for the seven test datasets. Our model demonstrates 23.5% reduction in the prediction time on average, due to the decrease in the number of support vectors from 9,822 to 7,809. This simpler model also achieves 7.6% decrease in test error, implying a more accurate distance function.

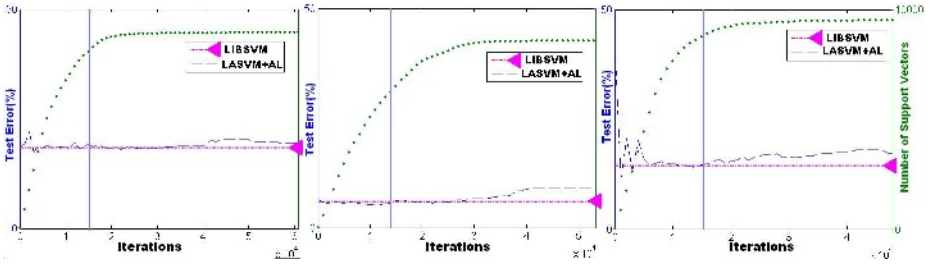


Fig. 3. Cross-validation on three-fold training datasets (from left to right: train[4,5] test[9]; train[4,9] test[5]; train[5,9] test[4]). The optimal iteration number for early stopping is shown with a vertical line, and the LIBSVM test error with a triangle.

4.2 Name Disambiguation Performance

We measure disambiguation performance at two levels as in [3]. At the pair level, **pairwise precision** pP is defined as the fraction of pairs in the same cluster being coreferent, **pairwise recall** pR as the fraction of coreferent pairs put into the same cluster, and **pairwise F1** $pF1$ as the harmonic mean of pP and pR. At the cluster level, **cluster precision** cP is the ratio of the number of completely correct clusters to the total number of clusters retrieved, whereas **cluster recall** cR is the portion of true clusters retrieved. Likewise, **cluster F1** $cF1$ is the harmonic mean. The **ratio of cluster size** RCS is defined as the number of clusters retrieved versus the number of true clusters. Note that cluster level metrics give no credits to clusters that miss some papers or are partially correct, making them more stringent and less telling than the pairwise metrics.

Table 3 shows the disambiguation accuracy of the entire system. Overall, it achieves 90.6% pairwise F1 metric, and 63.8% of the author name clusters are completely correct. The RCS is 0.944 (close to the optimal value 1.0), implying that the number of unique authors can be estimated with the number of clusters from disambiguation results. To test the efficiency, the entire CiteSeer metadata dataset is disambiguated in 3,880 minutes, yielding 418,809 unique authors.

5 Conclusion

An integrative framework is introduced to efficiently and adaptively resolve the name disambiguation problem. In this framework, a blocking module significantly

reduces the cost of similarity calculation. Our results show that with active sample selection and early stopping, learning a distance function is faster and more accurate than that of traditional SVM's. Our framework is easily expandable to the growing datasets. First, online setting enables the incorporation of new information without retraining the entire collection. Second, DBSCAN corrects the rare cases where the transitivity property is violated by merging or splitting clusters. We also formally prove the correctness of using DBSCAN for coreference resolution and the absence of transitivity problem for core points.

Table 3. Disambiguation accuracy

Dataset	pP	pR	pF1	cF1	RCS	Dataset	pP	pR	pF1	cF1	RCS
A. Gupta	0.914	0.960	0.937	0.483	0.977	J. Smith	0.815	0.853	0.834	0.625	0.860
A. Kumar	0.995	0.941	0.972	0.667	0.845	K. Tanaka	0.980	1.000	0.990	0.923	0.950
C. Chen	0.782	0.970	0.866	0.739	1.049	M. Jones	0.895	0.873	0.884	0.717	0.774
D. Johnson	0.761	0.948	0.844	0.434	1.024	M. Miller	0.775	0.953	0.855	0.451	1.028
J. Anderson	0.909	0.978	0.942	0.675	0.791	**Mean**	0.873	0.944	0.906	0.638	0.944
J. Robinson	0.908	0.963	0.935	0.667	1.143	**Std. Dev.**	0.085	0.046	0.056	0.150	0.122

Acknowledgments. This work was partially supported by grants from Microsoft Research and the National Science Foundation (NSF).

References

[1] Han, H., Giles, C.L., Manavoglu, E., Zha, H., Zhang, Z., Fox, E.: Automatic document metadata extraction using support vector machines. In: Proceedings of Joint Conference on Digital Libraries (JCDL 2003). (2003) 37–48

[2] McCallum, A., Nigam, K., Ungar, L.: Efficient clustering of high-dimensional data sets with application to reference matching. In: Proceedings of KDD. (2000)

[3] Wellner, B., McCallum, A., Peng, F., Hay, M.: An integrated, conditional model of information extraction and coreference with application to citation matching. In: Proceedings of the 20th Conference on Uncertainty in AI. (2004) 593–601

[4] Han, H., Zha, H., Giles, C.L.: Name disambiguation in author citations using a K-way spectral clustering method. In: Proceedings of JCDL. (2005) 334–343

[5] Han, H., Giles, C.L., Zha, H., Li, C., Tsioutsiouliklis, K.: Two supervised learning approaches for name disambiguation in author citations. In: Proceedings of Joint Conference on Digital Libraries (JCDL 2004). (2004) 296–305

[6] Lee, D., On, B., Kang, J., Park, S.: Effective and scalable solutions for mixed and split citation problems in digital libraries. In: ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS). (2005)

[7] Mann, G.S., Yarowsky, D.: Unsupervised personal name disambiguation. In: Proceedings of CoNLL-7. (2003) 33–40

[8] Bekkerman, R., McCallum, A.: Toward conditional models of identity uncertainty with application to proper noun coreference. In: IJCAI Workshop. (2003)

[9] Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd International Conference on Knowledge Discovery and Data Mining (KDD 1996). (1996) 226–231

[10] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name-matching in information integration. IEEE Intelligent System **18**(5) (2003)
[11] Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag (1995)
[12] Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast kernel classifiers with online and active learning. Journal of Machine Learning Research **6** (2005) 1579–1619
[13] Schohn, G., Cohn, D.: Less is more: Active learning with support vector machines. In: Proc. of 7th International Conf. on Machine Learning (ICML). (2000)
[14] Ankerst, M., Breunig, M., Kriegel, H., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. of ACM SIGMOD. (1999) 49–60
[15] Chang, C., Lin, C.: LIBSVM: a library for support vector machines. (2001)

Adaptive Segmentation-Based Symbolic Representations of Time Series for Better Modeling and Lower Bounding Distance Measures

Bernard Hugueney*

Universitè PARIS-DAUPHINE
LAMSADE Place du Maréchal de Lattre de Tassigny
75775 PARIS CEDEX 16
bernard.hugueney@lamsade.dauphine.fr

Abstract. Time series data-mining algorithms usually scale poorly with regard to dimensionality. Symbolic representations have proven to be a very effective way to reduce the dimensionality of time series even using simple aggregations over episodes of the same length and a fixed set of symbols. However, computing adaptive symbolic representations would enable more accurate representations of the dataset without compromising the dimensionality reduction. Therefore we propose a new generic framework to compute adaptive Segmentation Based Symbolic Representations (SBSR) of time series. SBSR can be applied to any model but we focus on piecewise constant models (SBSRL0) which are the most commonly used. SBSR are built by computing both the episode boundaries and the symbolic alphabet in order to minimize information loss of the resulting symbolic representation. We also propose a new distance measure for SBSRL0 tightly lower bounding the euclidean distance measure.

1 Introduction

Time series are easily collected in huge amounts only limited by the sampling rate of the sensors used to fill the databases. However, the sheer amount of available data prevents the direct analysis of those time series. Furthermore, even the automated data-mining algorithms have a hard time scaling up to the task handling long time series. For this reason, one usually computes new representations of the time series. There can be two (non exclusive) goals for those representations:

- Dimensionality reduction. The representations should preserve as much as possible the underlying information. Usually, such representations allow to reconstruct time series as close as possible to the original time series (according to some distance measure). The difference between those representations

* This work as been supported by grants from Région-Île-de-France.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 545–552, 2006.
© Springer-Verlag Berlin Heidelberg 2006

and the one used for compression is the need to enable some operations directly on those representation (for example computing distance measures).

– Information extraction. The representations should make the underlying information explicit. Those high-level representations must exhibit the relevant information as defined beforehand. This is akin to pattern matching and symbolic process-monitoring [10] where episodes of interest are detected and labeled according to a predefined set of shapes.

There has recently been a growing interest in symbolic representations of time series. The poster child of such representations being SAX (Symbolic Aggregate approXimation)[9] which enables many data-mining tasks. Symbolic representations fall clearly into the information extraction kind of time series representations, where episodes are associated to symbols interpretable according to the data-mining task. In this paper, we propose to learn those symbols without prior knowledge in order to build more relevant symbolic representations. The rest of this paper is organized as follows. Section 2 presents background and related work in time series symbolic representations and adaptive versus non adaptive representations. Section 3 presents our proposed Segmentation-Based Symbolic Representations (SBSR) with an emphasis on those based on piecewise constant segmentations (with linear model of order 0) SBSR-L0. In the same section, we propose a generic algorithm to compute SBSR and a specific algorithm for SBSR-L0. In sections 4, we evaluate the modeling accuracy of our algorithm. For space constraint reasons, we could only illustrate the results on our real-world dataset in this paper. However, more extensive evaluations, along with more detailed algorithms, are available in an expanded version of this paper [2]. In section 5 we discuss the distance measures that can be defined over SBSR in order to take advantage of the accurate modeling for a better lower bounding distance measure. We present our conclusions and current research directions in section 6.

2 Background and Related Works

As we have seen in section 1, time series databases are usually much too large to be tackled directly by data-mining algorithms. As far as this paper is concerned, the problem does not come from the number of time series in the database, but from the length of the time series. For this reason, we build representations of time series in order to reduce the dimensionality of the dataset, as it is a very common preprocessing step. A model for time series databases handling both the number of time series and the length of those time series is presented in [4]. Many time series data-mining algorithms are based on a distance measure between time series, so defining such a distance measure is crucial when designing new time series representations, as we will see in section 5. Amongst the many time series representations already available, we will focus on those tackling issues the most related to our need for concise and expressive time series representations:

1. SAX (Symbolic Aggregate approXimation), for the information extraction process of turning numerical time series into strings of symbols,
2. APCA (Adaptive Piecewise Constant Aggregation), for the advantages of locally adaptive representations with regard to conciseness / modeling accuracy tradedoff, when compared to its non-adaptive counterpart (Piecewise Aggregate Approximation).

The dataset of the univariate time series to be represented is defined as follows: Let $S = \{TS^i\}_{i\in\{1...M\}}$ be the dataset of M time-series. Each of the time series is defined on the same time stamps: $TS^i = \{(v_j^i, d_j)\}_{i\in\{1...N\}}$ with $v_j^i \in \mathbb{R}$ and $d_j \in D$, D being the temporal definition domain of the time series.

2.1 SAX: Symbolic Aggregate Approximation

The underlying principles of SAX are that the values taken by the normalized time series follow a normal distribution, and that the time series are "oversampled" according to the interesting patterns. Amongst the advantages of such a simple representation, is the fact that computational requierments for building such representations are minimal and that it is easy to compare representations of different time series if the episodes are of the same length. Another important asset of SAX is the distance measure that can be defined to lower bound the euclidean distance. SAX is based on the mean value of the time series over episodes of the same length. The sequence of those means is in fact another time series representation, but a numerical one, called Piecewise Aggregate Approximation (PAA).

2.2 Adaptive Piecewise Constant Approximation Versus Piecewise Aggregate Approximation

SAX is a kind of symbolic version of PAA, that is based on piecewise constant models (or linear models of order 0) over episodes of the same duration, built by quantizing the means according to a partition. PAA was introduced as an effective representation of time series in [7] where it was shown to enable dimensionality reduction and indexing of time series databases. As always, there is a tradeoff to be made between the accuracy of the representation and its conciseness. The accuray is often measured by the sum of square errors (SSE) and the conciseness by the complexity of the representation (i.e. the number of episodes for PAA).

Better accuracy/conciseness tradeoffs can be achieved thanks to APCA([5]), an adaptive variant of PAA where episode length is locally adapted to the values taken by the time series. In search of time series symbolic representations that would yield the benefits of adaptive representations, that is better conciseness versus accuracy tradeoffs, we propose a new kind of symbolic representations based on segmentations.

3 SBSR: Segmentation-Based Symbolic Representations

3.1 Generic Framework for Symbolic Representations of Time Series

With the notations presented in section 2, we defined the symbolic representations in [1] as follows:

- $E = \{e_p = (d_p, d_{p+1})\}_{l \in \{1...P\}}$ is a set of P episodes that is a partition of the time domain D.
- Λ is a set of K symbols.
- $SR(TS) = \{(e_p, \lambda_p, \ldots)\}_{p \in \{1...P\}}$ is a symbolic representation of TS that allows to define a function : $E \rightarrow \Lambda, e_p \mapsto \lambda_p$. Each tuple of $SR(TS)$ is at least made of an episode and a symbol, but other information can be added, either numerical of symbolical in nature.

This very broad definition of symbolic representations encompasses a great number of symbolic representations. Different kinds of such symbolic representations are studied in [3], but we now present a new kind of Segmentation-Based Symbolic Representations (SBSR) that makes use of segmentation algorithms in order to compute the sets of episodes.

3.2 Segmentation-Based Symbolic Representations

Segmented models of time series can be defined as $Seg(TS) = \{(e_p, \phi_p)\}$ where ϕ_p is a set of parameter values for the given model. PAA and APCA are both segmented models for linear models of order 0, and linear models of order 1 are used in [6].

SBSR generalizes the quantization of the means in PAA that leads to SAX by defining clusters of parameters values. Let $\Lambda = \{\lambda_k\}_{k \in \{1...K\}}$ be the alphabet of symbols associated to $\Gamma = \{\gamma_k\}_{k \in \{1...K\}}$ the partition of the parameter space, $SBSR(TS) = \{(e_p, \lambda_p, \ldots)\}_{p \in \{1...P\}}$ is a symbolic representation such that $e_p \mapsto \lambda_k / \phi_p \in \gamma_k$ for a given segmentation of TS into P segments $Seg(TS) = \{(e_p, \phi_p)\}_{p \in \{1...P\}}$.

SBSR can be applied to any model but we now focus on the representation based on linear models of order 0, that we call SBSR-L0.

3.3 Segmentation-Based Symbolic Representations with Linear Models of Order 0

SBSR-L0 are defined as follows:

- $\Lambda = \{\lambda_k\}_{k \in \{1...K\}}$ is the alphabet of symbols. Each λ_k is related to a prototypical numerical value $\phi'_k \in \mathbb{R}$ and an interval $I_k = \{\phi \in \mathbb{R} / (\phi - \phi'_k)^2 < (\phi - \phi'_l)^2 \forall l \neq k\}$.
- $E = \{e_p = (d_p, d_{p+1})\}_{p \in \{1...P\}}$ is the partition of D into P episodes.
- SBSR-L0$(TS) = \{(e_p, \lambda_p)\}_{p \in \{1...P\}}$, where λ_p is the symbol relating to the interval that contains $\overline{TS}[e_p]$, the mean value of TS over the episode e_p.

What we call *restricted segmentation* is a segmentation where the model parameters can only take values from a finite subset of the parameter space (here $\{\phi'_k\}_{k\in\{1...K\}}$). Fig.1 shows the SBSR-L0 representation of the daily extract of the time series. It is important to note that the levels used for this extract are computed in order to represent the whole time series and not only this small extract, hence it is a daily extract of the SBSR-L0 representation of the whole time series and not SBSR-L0 representation of a daily extract.

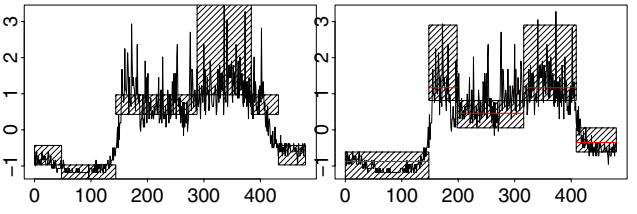


Fig. 1. Daily extract (over 480 points) of SAX and SBSR-L0 representations with 10 symbols

As we have seen in section 3.1 the most difficult part is often to devise efficient algorithms to compute E and Λ in order to maximize the modeling accuracy. However, it is possible to take advantage of the efficient segmentation algorithms to compute SBSR in general and especially SBSR-L0.

3.4 Generic Algorithm for Computing SBSR

The generic algorithm that we propose is a two-step iterative process much like the k-means[11] and E-M algorithms:

1. Compute initial segmentation $Seg^0(TS) = \{({e_p}^0, {\phi_p}^0)\}_{p\in\{1...P\}}$ of TS into P segments with the best algorithm affordable according to the number of points N.
2. Optimization of the interpretations of alphabet $\{{\phi'_k}^s\}$ for a given set of episodes computed in previous step s. We cluster the set of extracts of $\{TS[e_p^s]\}$ into K clusters according to the clustering criterion and compute prototypical values $\{\phi_k^s\}$ that minimize the reconstruction error when each episode is associated to the prototypical value of its cluster.
3. Optimization of the set of episodes E^{s+1} according to a given set of interpretations $\{{\phi'_k}^s\}$. We perform restricted segmentation of TS with model parameters values restricted to $\{{\phi'_k}^s\}$.
4. Repeat from step 2 until convergence

4 Modeling Accuracy

In order to provide simple distance measures between the time series of our datasets, we decided to compute only one set of episodes E and one set of symbols

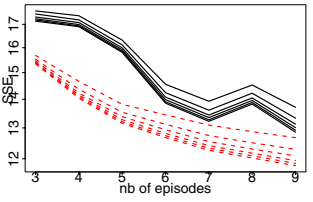


Fig. 2. SSE for numerical reconstructions from SAX and SBSR-L0 representations of our real-world dataset

and their interpretations for the whole dataset, and not specific sets for each of the time series. The modeling accuracy would have been even better with specific sets of episodes and alphabets of symbols. Given that SAX representations do not define a numerical reconstruction, we decided to assign to each SAX symbol a numerical value that is the mean value of the normal distribution over the associated interval.

The real-world dataset that we used to evaluate SBSR-L0 is based on urban traffic information sampled every 3 minutes by 423 sensors. The time series are normalized and the dataset is made of the 5000 daily extracts of 480 points from those time series. With notations in section 2, we have $M = 5000$ and $N = 480$. In Fig.2, we present the SSE for numerical reconstructions from SAX and SBSR-L0 representations. Each curve represents the SSE for a number of symbols ($K \in \{4 \dots 9\}$. It is important to notice that the SSE scale is logarithmic. The dahsed lines (resp. plain lines) are all thoses corresponding to SBSR-L0 (resp. SAX) representations.Within each of these groups the better results are of course those of the representations with the most symbols. It is interesting to note that SAX modeling is not always increasingly more accurate when more episodes are used for the representations.

5 Distance Measures

As we have seen in section 1, in order to be useful for data-mining tasks, representations of time series must enable the use of distance measures. Of course, the main advantage of symbolic representations is to enable specific manipulations using the domain semantic. A semantic distance between symbolic representations of time series would make use of an edit distance (or Levenshtein distance [8]) between the symbol strings and a semantic distance between each symbol.

However, it is often very important to be able to compare time series according to the underlying numerical data, for example by using the euclidean distance.

As we have just seen, the distance factor between symbols of SAX representations is based on the theoretical worst-case of mean values over the set of represented time-series extract. This pessimistic approach is good because it does not require any information from the represented time series and is not dependent on the datasets. However, it would be possible to lower bound more efficiently the euclidean distance with information about the mean values actually taken

by the time series in the dataset. This also makes the case for representations both numerical and symbolical in nature. As for numerical computations such as distance measures, one cannot beat the accuracy of numerical information. From the smallest to the largest amount of numerical information that can be put into SBSR-L0, we have:

1. No additionnal information, as in SAX. Let us call the associated distance D_{SAX}.
2. Dataset global information. For each symbol, we store the actual minimal and maximal values of mean values taken by the time series extracts over episodes represented by the symbol. Let us call the associated distance D_g.
3. Time series local information. For each symbol and each of the time series, we store the actual minimal and maximal values of mean values taken by the time series extracts over episodes represented by the symbol. Let us call the associated distance D_l.
4. Episode local information, as in APCA. Let us call the associated distance D_{APCA}.

It is obvious that we have $D_{SAX}(TS_{i1}, TS_{i2}) \leq D_g(TS_{i1}, TS_{i2}) \leq D_l(TS_{i1}, TS_{i2}) \leq D_{APCA}(TS_{i1}, TS_{i2}) \leq D(TS_{i1}, TS_{i2})$, $\forall (TS_{i1}, TS_{i2}) \in S \times S$.

The best tradeoff between the space requirements and efficient lower bounding is application dependent. In Fig.3, we show the results of the evaluation of the distance functions over our real-world dataset (presented in section 4)for various values of K and P.

We computed the mean thightness for all of the distances between each pair of time series in the dataset. We represented the tightness of 4 distance measures ($D_{SAX}, D_g, D_l, D_{APCA}$) on both SAX and SBSR-L0 representations. That should be 8 curves but we can see only 6 because D_{SAX} and D_g are extremely close to each other.

From the lowest (worst) to the highest (best), the curves that we can see are :

1. D_{SAX} and D_g for SAX
2. D_{SAX} and D_g for SBSR-L0
3. D_l for SAX
4. D_l for SBSR-L0
5. D_{APCA} for SAX
6. D_{APCA} for SBSR-L0

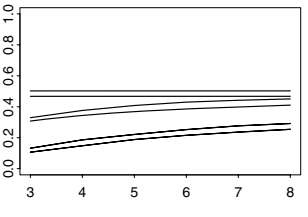


Fig. 3. Mean tightness of lower bounding by the various distance measures for 9 symboles and 3 to 10 episodes on our real-world dataset

6 Conclusions and Future Works

We have proposed SBSR, a new generic framework for locally symbolic and numeric/symbolic representations of time series based on segmentations and a generic algorithm to build those representations.

Amongst SBSR, we focused on those based on piecewise constant models with prototypical values associated to symbols: SBSR-L0. We proposed an algorithm to build locally optimal SBSR-L0 and made experimental validations of the modeling accuracy of SBSR-L0 on a real-world dataset. We proposed various distance measures between SBSR-L0 representations, corresponding to different tradeoffs between space requirements and accuracy of the euclidean distance lower bounding and provided experimental results for those distances measures.

The SBSR framework allows to define numerous kinds of symbolic or numeric/symbolic representations of time series. When the information to extract concerns the local linear trends of time series, we ressort to SBSR based on clustering of trends in linear models of order 1 (SBSR-L1-T).

References

1. G. Hebrail and B. Hugueney. Symbolic representation of long time series. In Conference on Applied Statistical Models and Data Analysis (ASMDA), pages 537-542, June 2001.
2. B. Hugueney. Expanded version of Adaptive Segmentation-Based Symbolic Representations of Time Series for BetterModeling and Lower Bounding DistanceMeasures. http://www.lamsade.dauphine.fr/hugueney/PKDD2006-Expanded.pdf.
3. B. Hugueney. Représentations symboliques de longues series temporelles. PhD thesis, LIP6, 2003.
4. B. Hugueney, G. Hébrail, and Y. Lechevallier. Computing summaries of time series databases with clustering and segmentation. Int. Fed. of Classification Societies, 2006.
5. E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. SIGMOD Record (ACM Special Interest Group on Management of Data), 30(2):151-162, June 2001.
6. E. Keogh and M. J. Pazanni. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, Proceedings of the Forth International Conference on Knowledge Discovery and Data Mining (KDD- 98). AAAI Press, 1998.
7. E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems Journal, 2000.
8. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady., 10(8):707-710, Feb. 1966. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.
9. J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, pages 2-11. ACM Press, 2003.
10. P. L. Love and M. Simaan. Automatic recognition of primitive changes in manufacturing process signals. Pattern Recognition, 21(4):333-342, 1988.
11. J. MacQueen. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281-297, 1967.

A Feature Generation Algorithm for Sequences with Application to Splice-Site Prediction

Rezarta Islamaj[1], Lise Getoor[1], and W. John Wilbur[2]

[1] Computer Science Department, University of Maryland, College Park, MD 20742
[2] National Center for Biotechnology Information, NLM, NIH, Bethesda, MD 20894
{rezarta, getoor}@cs.umd.edu, wilbur@ncbi.nlm.nih.gov

Abstract. In this paper we present a new approach to feature selection for sequence data. We identify general feature categories and give construction algorithms for each of them. We show how they can be integrated in a system that tightly couples feature construction and feature selection. This integrated process, which we refer to as *feature generation*, allows us to systematically search a large space of potential features. We demonstrate the effectiveness of our approach for an important component of the gene finding problem, splice-site prediction. We show that predictive models built using our feature generation algorithm achieve a significant improvement in accuracy over existing, state-of-the-art approaches.

1 Introduction

Many real-world data mining problems involve data best represented as sequences. Sequence data comes in many forms including: 1) human communication such as speech, handwriting and language, 2) time sequences and sensor readings such as stock market prices, temperature readings and web-click streams and 3) biological sequences such as DNA, RNA and protein. Sequence data in all domains contains useful 'signals', features, that enable the correct construction of classification algorithms.

Extracting and interpreting the features is known to be a hard problem. In many cases a brute force approach is taken, in which the sequence classification models are provided with a huge number of features in the hope that the important features are not overlooked. The large number of features introduces a dimensionality problem which has several disadvantages. First, enumerating all possible features is impractical. Second, many features are irrelevant to the classification task and have an adverse effect on accuracy. And third, knowledge discovery becomes difficult because of the large number of parameters involved.

The focus of this paper is on a scalable method for feature generation for sequences. We present an algorithm that explores the space of possible features and identifies the most useful ones. Our focused **feature generation algorithm (FGA)** integrates feature construction and feature selection in a systematic way. Our method is scalable because it incrementally generates more complex features from currently selected ones.

We validate our method on the task of splice-site prediction for pre-mRNA sequences. Splice sites are the locations in the DNA sequence, which are boundaries for protein coding and non-coding regions. Accurate location of splice sites is an important component in the gene finding problems. It is a particularly difficult problem since

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 553–560, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the sequence characteristics, i.e. pre-mRNA sequence length, coding sequence length, number of exons and their lengths, and interrupting intron sequence lengths do not follow any known pattern, making it hard to locate the genes.

We demonstrate the effectiveness of our approach by comparing it with a state-of-the-art method, GeneSplicer. Our predictive models show significant improvement in accuracy. Our final feature set, achieves a 6.3% improvement in the 11-point average precision when compared to GeneSplicer. At the 95% sensitivity level, our method yields a 50% improvement in specificity. Our contribution is two-fold. First, we give a general feature generation framework appropriate for any sequence data problem. Second, we provide new results and identify a set of features for splice-site prediction that should be of great interest to the gene-finding community.

2 Related Work

Feature selection techniques have been studied extensively in text categorization [1, 2, 3,4,5]. Recently they have begun receiving more attention for applications to biological data. A good introduction for filtering methods in the prediction of translation initiation sites is given in [6]. Various feature selection techniques for prediction of splice sites have been studied in [7, 8, 9]. And in [10], SpliceMachine is described with compelling results. In addition, there is a significant amount of work on splice-site prediction. One of the most well-known approaches is GeneSplicer proposed by Pertea et al [11].

3 Data Description

We validate our methods on a dataset which contains $4,000$ RefSeq[1] pre-mRNA sequences. In a pre-mRNA sequence, a human gene is a protein coding sequence which is characteristically interrupted by non-coding regions, called introns. The coding regions are referred to as exons. The acceptor splice site marks the start of an exon and the donor splice site marks the end of an exon. All the pre-mRNA sequences in our dataset follow the AG consensus for acceptors and GT consensus for donors.

We focus on the prediction of acceptor splice sites which is considered to be a harder problem. Following the GeneSplicer format, we mark the splice site and take a subsequence consisting of 80 nucleotides upstream from the site and 80 nucleotides downstream. We construct negative examples by choosing random AG-pair locations that are not acceptor sites and selecting subsequences as we do for the true acceptor sites. Our data contains 20,996 positive instances and 200,000 negative instances.

4 Feature Generation

The feature types that we consider capture compositional and positional properties of sequences. These apply to any sequence data defined over some fixed alphabet. For each feature type we describe an incremental feature construction procedure. The feature construction starts with an initial set of features and produces an expanded set of features. Incrementally, it produces richer, more complex features for each iteration.

[1] http://www.ncbi.nlm.nih.gov/RefSeq/

4.1 Feature Types and Construction Procedures

Compositional features. A *general k-mer* is a string of k-characters. This feature type is useful for capturing information like coding potential and composition in the sequence.

Construction Method. Given an initial set of k-mer features, this construction method expands them to a set of $(k+1)$-mers by appending the letters of the alphabet to each k-mer feature.

Region-specific compositional features. Splice-site sequences characteristically have a coding region and a non-coding region. For the acceptor splice-site sequences, the region of the sequence on the left of the splice-site position (upstream) is the non-coding region, and the region of the sequence from the splice-site position to the end of sequence (downstream) is the coding region. These regions may exhibit distinct compositional properties. In order to capture these differences we use *region-specific k-mers*.
Construction Method. The construction procedure of upstream and downstream k-mer features is the same as the general k-mer method, with the addition of region indicator.

Positional features. Position-specific nucleotides are the most common features used for finding signals in the DNA stream data [12]. These features capture the correlation between different nucleotides and their relative positions. The *position specific k-mers* capture the correlations between k-adjacent nucleotides. At each position i in the sequence these features represent the substring appearing at positions $i, i+1, .., i+k$.
Construction Method. This construction method starts with an initial set of position-specific k-mer features and extends them to a set of position-specific $(k+1)$-mers by appending the letters of the alphabet to each position-specific k-mer feature.

Conjunctive positional features. To capture the correlations between different nucleotides in nonconsecutive positions in the sequence, we propose *conjunctive position-specific features*. We construct these complex features from conjunctions of basic position-specific features. The dimensionality of this kind of feature is inherently high.

Construction Method. Given an initial set of k-conjuncts, this construction method selects from the set of basic position-specific features to add another conjunct in an unconstrained position, therefore constructing the set of $(k+1)$-conjuncts.

4.2 Feature Selection

Feature selection methods reduce the set of features by keeping only the useful features for the task at hand. The problem of selecting useful features has been the focus of extensive research and many approaches have been proposed [1, 2, 3, 4, 5].

In our experiments we consider several feature selection methods to reduce the size of our feature sets. We use several filter approaches including *Information Gain (IG), Chi-Square (CHI), Mutual Information (MI)*, *KL-distance (KL)* for initial pruning of feature types sets during the generation stage. Due to space limitations, in the experiments section, we present the combination that produced the best results. In our data,

we found that mutual information performed best for selecting compositional features, chi-squared for positional features and information gain for conjunctive features. At the final collection step, we combine this with an embedded method based on recursive feature elimination [9] used in our final feature collection stage. The weights w_i of the decision boundary of a linear SVM can be used as feature weights to derive feature ranking. We use the C-Modified Least Squares (CMLS) classifier [13] and refer to this method as W-CMLS. We recursively train the classifier and remove low scoring features.

4.3 Feature Generation Algorithm (FGA)

The traditional feature selection approaches consider a single brute force selection over a large set of all features of all different types. By categorizing the features into different feature types we can apply appropriate construction and selection methods suitable to the different types. Thus we can extract relevant features from each feature type set more efficiently than if a singe selection method had been applied to the whole set. We use the following algorithm:

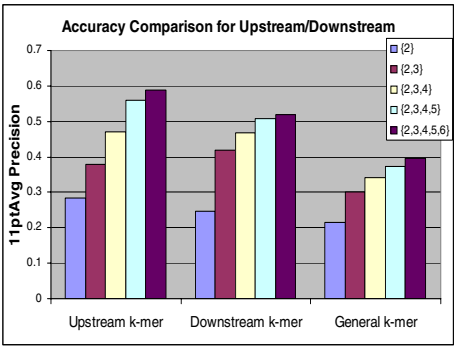
- *Feature Generation.* The first stage generates feature sets for each feature type. For each defined feature type, we tightly couple the corresponding feature construction step with a specified feature selection step. We iterate through these steps to generate richer and more complex features. During each iteration, we eliminate features that are assigned a low selection score by the feature selection method.
- *Feature Collection and Selection.* In the next stage, we collect the features of different types and apply another selection step.
- *Classification.* The last stage of our algorithm builds a classifier over the final set of features.

For this type of problem it is not unusual to spend a lot of computational resources, especially in the training phase. While feature generation remains a computationally intensive process, the organization of the generation process according to the different types allows us to search a much larger space efficiently. For the time complexity of the classification algorithm, we use CMLS which is very efficient. In addition, this feature generation approach has other advantages such as the flexibility to adapt with respect to the feature type and the possibility to incorporate the module in a generic learning algorithm.

5 Experimental Results for Splice-Site Prediction

We conducted a wide range of experiments using a variety of classifiers, and here we present a summary of them. We present results for the classifier that consistently gave the best results, CMLS.

We use the *11-point average* (11ptAvg) [14] to evaluate the performance of our algorithm. For any recall ratio, we calculate the precision at the threshold which achieves that recall ratio and compute the average precision. The 11ptAvg is the average of precisions estimated at recall values 0%, 10%, 20%, ., 100%. The ability of our algorithm to discriminate true acceptor site sequences from normal sequences is evaluated also using Receiver Operating Characteristic (ROC) curve analysis Another performance



(a)

No. features	IG	MI	Chi	KL
9000	87.44	87.39	87.41	87.44
8000	87.40	86.79	87.35	87.33
7000	87.14	85.24	87.15	87.19
6000	86.51	82.92	86.51	86.68
5000	85.93	80.33	85.94	85.89
4000	83.69	75.46	83.57	83.87
3000	82.71	66.63	82.50	82.97
2000	76.64	67.49	80.28	68.65

(b)

Fig. 1. (a) Comparison between different feature type sets performances, upstream k-mers, downstream k-mers, and general k-mers shown for different k (b) 11ptAvg precision results for FGA varying the feature set size of position-specific collection of k-mers through different feature selection methods

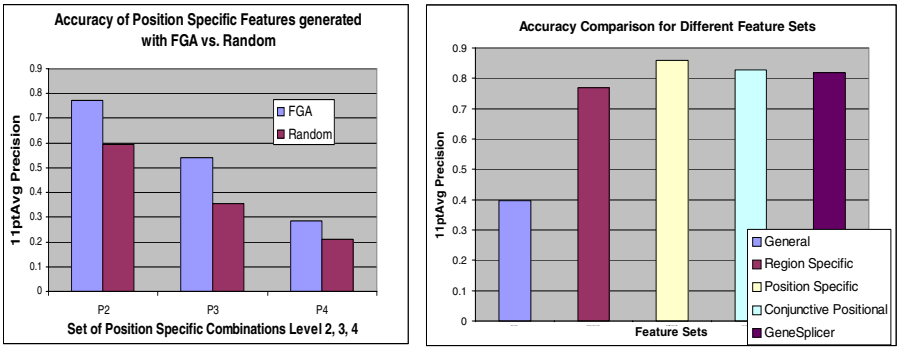
measure commonly used for biological data is the *false positive rate(FPr)* defined as $FPr = \left(\frac{FP}{FP+TN}\right)$ where FP, and TN are the number of false positives and true negatives respectively. FPr can be computed for all recall values by varying the decision threshold of the classifier. We also present results using this measure. In all our experiments, the results reported use three-fold cross-validation.

5.1 Accuracy Results of FGA

We begin with a brief evaluation of the effectiveness of the different feature types used in isolation.

Compositional features and region-specific compositional features. We examine each k-mer feature set independently for each value of k from 2 to 6. Figure 1(a) shows the accuracy results for the region-specific k-mers and the general k-mer feature sets as we collect them after each iteration. In our experiments, *MI* selection method worked best for compositional features. We notice that k-mer features carry more information when they are associated with a specific region (upstream or downstream) and this is shown by the significant increase in their 11ptAvg precisions.

Positional features. Next, we examine each *position-specific k-mer* feature set independently. We explore k-values from 1 to 6. The prediction results for this feature type (data not shown) after each generation step gradually increase until level 3, then gradually drop. This can be explained with the exponential increase in the number of features after each level. In Figure 1(b), we use feature selection to have a mix of position-specific k-mers for k values from 1 to 3. This table shows results of repeated selection for *IG, MI, CHI* and *KL* feature selection methods. Of these, *CHI* retains the highest precision among the four methods. Our paired-t tests for statistical significance reveal that these values although similar are statistically significant.



(a) (b)

Fig. 2. a) 11ptAvg results for the position specific feature sets generated with FGA algorithm vs randomly generated features. b) Performance results of the FGA method for different feature types as well as the GeneSplicer program.

Conjunctive positional features. Finally, we examine conjunctive positional features. The number of these features grows exponentially and it is clearly very cumbersome to test for relevance more than 40 million unique combinations of triple conjuncts. We explore sets of 2 to 4 conjuncts denoted as $(P2, P3, P4)$. We use the *IG* selection method to select the top scoring $1,000$ features and repeat the generation on the selected set to get the next level. In Figure 2(a), we show the performances of the conjunctive feature sets. For comparison, we introduce a baseline method, which is the average of 10 trials of randomly picking $1,000$ conjunctive features from each level.

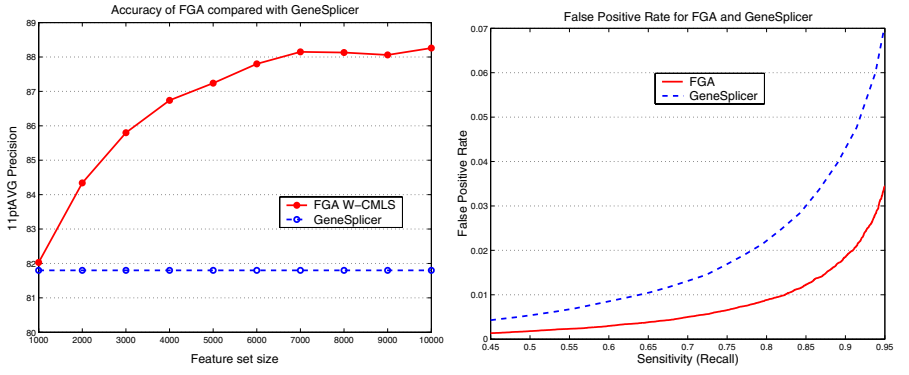
Summary. Next, we compare collections of different levels of the feature sets of different types. The results are summarized in Figure 2(b).

Compositional features and region-specific compositional features. The first two bars show the results for the best $2,000$ k-mer features for k ranging from 2 to 6. General k-mers result in an 11ptAvg of only 39.84%, while the result of the combined upstream and downstream k-mer features is 77.18%.

Position-specific k-mers. The third bar shows the results for $5,000$ position-specific k-mer features selected using the *CHI* selection method for k ranging from 1 to 3. The 11ptAvg precision is 85.94%.

Conjunctive positional features. The next bar shows the results for a collection of $3,000$ conjunctive positional features for k ranging from 1 to 4 selected using *IG*. The 11ptAVG precision that this collection set gives is 82.67%. These results clearly show that using complex position-specific features is beneficial. Interestingly, these features typically are not considered by existing splice-site prediction algorithms.

Figure 2(b) also shows the performance of GeneSplicer on the same dataset. We see that even in isolation, our position-specific features and our conjunctive positional features perform better than GeneSplicer. These results are also statistically significant.



(a) (b)

Fig. 3. (a) 11ptAvg precision results for FGA varying the feature set size, compared to GeneSplicer (b) The false positive rate results for FGA varying the sensitivity threshold, compared to GeneSplicer

Results using the full feature-type collection. In the following set of experiments, we show the results after we collect all the features that we have generated. We run our CMLS classification algorithm with a total feature set of size $10,000$ containing general k-mers, upstream/downstream k-mers, position-specific k-mers and conjunctive position-specific features. We achieve an 11ptAvg precision performance of 88.20%. This compares quite favorably with one of the leading programs in splice-site prediction, GeneSplicer, which yields an accuracy of 81.89% on the same dataset. The precision results at all individual recall points (data not shown) are consistently higher than those of GeneSplicer. In Figure 3(a) we explore more aggressive feature selection options and see that smaller feature sets of even $2,000$ outperform GeneSplicer significantly. In these experiments it is the more expensive W-CMLS selection method that we use in order to select a smaller working feature set.

We present the false positive rates for various recall values in Figure 3(b). Our feature generation algorithm, with its rich set of features, consistently performs better than GeneSplicer. Our false positive rates are favorably lower at all recall values. At a 95% sensitivity rate the FPr decreased from 6.92 to 3.44%. This significant reduction in false positive predictions can have a great impact when splice-site prediction is incorporated into a gene-finding program. In terms of AUC score FGA gives a result of 0.971 compared to GeneSplicer of only 0.935. It should also be noted that there is no significant difference in the running time of FGA compared to GeneSplicer. FGA performs a linear search (in terms of sequence length) along the given sequence in search for high scoring sites.

6 Conclusions

We presented a general feature generation framework which integrates feature construction and feature selection in a flexible manner. We showed how this method can be used to build accurate sequence classifiers. We presented experimental results for the

problem of splice-site prediction. We were able to search over an extremely large space of feature sets effectively, and we were able to identify the most useful set of features of each type. By using this mix of feature types, and searching over combinations of them, we were able to build a classifier which achieves an accuracy improvement of 6.3% over an existing state-of-the-art splice-site prediction algorithm. The specificity values are consistently higher for all sensitivity thresholds and the false positive rate has favorably decreased. In future work, we plan to apply our feature generation algorithm to more complex feature types and other sequence prediction tasks, such as translation start site prediction.

Acknowledgments

This research was supported in part by an appointment to the National Center of Biotechnology Information Scientific Visitors Program sponsored by the National Library of Medicine and administered by the Oak Ridge Institute for Science and Education.

References

1. Kohavi, R., John, G.: The wrapper approach. In: *Feature Extraction, Construction and Selection : A Data Mining Perspective*, Liu,H.,Motoda,H.,eds. Kluwer Academic Publishers (1998)
2. Koller, D., Sahami, M.: *Toward optimal feature selection*. In: ICML. (1996) 284–292
3. Yang, Y., Pedersen, J.: *A comparative study on feature selection in text categorization*. In: ICML. (1997)
4. Yu, L., Liu, H.: *Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution*. In: ICML. (2003)
5. Blum, A., Langley, P.: *Selection of relevant features and examples in machine learning*. Artificial Intelligence (1997)
6. Liu, H., Wong, L.: *Data mining tools for biological sequences*. Journal of Bioinformatics and Computational Biology (2003)
7. Degroeve, S., Baets, B., de Peer, Y.V., Rouze, P.: *Feature subset selection for splice site prediction*. In: ECCB. (2002) 75–83
8. Yeo, G., Burge, C.: *Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals*. In: RECOMB. (2003)
9. Zhang, X., Heller, K., Hefter, I., Leslie, C., Chasin, L.: *Sequence information for the splicing of human pre-mRNA identified by support vector machine classification*. Genome Research **13** (2003) 2637–2650
10. Degroeve, S., Saeys, Y., Baets, B.D., Rouz, P., de Peer, Y.V.: *SpliceMachine: predicting splice sites from high-dimensional local context representations*. Bioinformatics **21** (2005) 1332–1338
11. Pertea, M., Lin, X., Salzberg, S.: *GeneSplicer: a new computational method for splice site prediction*. Nucleic Acids Research **29** (2001) 1185–1190
12. Zhang, M.: *Statistical features of human exons and their flanking regions*. Human Molecular Genetics **7** (1998) 919–932
13. Zhang, T., Oles, F.: *Text categorization based on regularized linear classification methods*. Information Retrieval **4** (2001) 5–31
14. Witten, I., Moffat, A., Bell, T., eds.: *Managing Gigabytes*. 2 edn. Van Nostrand Reinhold (1999)

Discovering Image-Text Associations for Cross-Media Web Information Fusion

Tao Jiang and Ah-Hwee Tan

School of Computer Engineering
Nanyang Technological University, Nanyang Avenue, Singapore 639798
{jian0006, asahtan}@ntu.edu.sg

Abstract. The diverse and distributed nature of the information published on the World Wide Web has made it difficult to collate and track information related to specific topics. Whereas most existing work on web information fusion has focused on multiple document summarization, this paper presents a novel approach for discovering associations between images and text segments, which subsequently can be used to support cross-media web content summarization. Specifically, we employ a similarity-based multilingual retrieval model and adopt a vague transformation technique for measuring the information similarity between visual features and textual features. The experimental results on a terrorist domain document set suggest that combining visual and textual features provides a promising approach to image and text fusion.

1 Introduction

The diverse and distributed nature of the information on the World Wide Web has made it difficult to collate and track information related to specific topics. Techniques for web information fusion, involving filtering of redundant information, collating of information according to themes, and generation of coherent presentation, are needed for information users. As a useful technique for information fusion, document summarization has been discussed in a large body of literatures. Most document summarization methods however focus on summarizing *text* documents. As an increasing amount of non-text content, namely images, video, and sound, is becoming available on the web, summarizing multimedia information has posed a key challenge in web information fusion.

In this paper, we focus on one of the important problems in multimedia fusion, namely the extraction of association between multimedia components, in particular, images and texts. Our approach is consistent with those found in the literatures of hypermedia authoring and cross-document text summarization, that understanding the interrelation between information blocks are essential for collating information and generating final presentations.

By extending a process for multi-document summarization [1], we present a procedure for web document fusion (Figure 1) consisting of five stages as follows.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 561–568, 2006.
© Springer-Verlag Berlin Heidelberg 2006

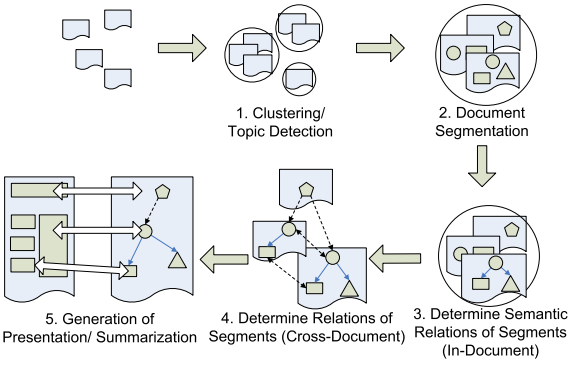


Fig. 1. An overview of the web information fusion process

1. The raw web documents are first clustered according to their topics.
2. Each document is divided into several atomic segments (atomic description unit) according to sub-topics and media types.
3. For each document, the relations among document segments are determined. In our work, we focus on the relations across media types.
4. Within a document cluster, the cross-document relations among document segments are determined. Duplicated contents are detected.
5. The document segments are reorganized and presented according to a summarization template and the in-document and cross-document relations.

We see that techniques developed for text documents can be used in the first two stages of the cross-media summarization process, i.e. document clustering and segmentation (where each multimedia object itself can be seen as a segment). For detecting the associations and relationships between multimedia components (e.g. text segments and images) within or across documents, we present a textual-visual vague transformation technique, borrowed from the field of multilingual retrieval [2], for extracting associations between images and texts from news web documents. The extracted image-text associations can be subsequently used for the third, fourth, and fifth stages of the summarization.

Note that our method is different from the existing efforts on image indexing using statistical modelling approaches originally proposed in the field of natural language processing [3]. Image indexing tends to establish the correspondence between keywords (concepts) and particular image regions. Our task, however, does not require such a correspondence between the contents of the text segments and the associated images. In the next two sections, we present our methods for data preprocessing and image-text association learning. Section 4 reports our experiments. Concluding remarks are given in Section 5.

2 Harvesting and Preprocessing of Texts and Images

We develop an image crawler, named "ICrawl", based on Yahoo Search API. Upon receiving a query, ICrawl searches images through Yahoo search engine,

downloads the images retrieved, and extracts the textual contents from the web pages wherein the images appear. The extracted textual contents include the tips and captions of the images, the keywords extracted from the URLs/tips/captions, and long text paragraphs (more than 15 words).

2.1 Textual Feature Extraction

Currently, we treat each text paragraph extracted from web pages as a text segment. We tokenize the text segments, add part-of-speech tags, remove stop words, replace tokens with their stems, filter out terms with unwanted POS tags (only nouns, verbs and adjectives are left), and finally generate term vectors. For images downloaded from the web, their surrounding texts, including captions, tips, and keywords in URL, are extracted. Like text segments, the extracted surrounding texts are processed to form term vectors.

For calculating the term weights of the term vectors, we use a model, named TF-ITSF, similar to a traditional TF-IDF model. For a text segment or an image text description (the surrounding text of an image) ts in a web document d, we use the following equation to weight a term w:

$$w^d(ts) = tf(ts, w) \cdot \log \frac{N^d}{tsf^d(w)} \tag{1}$$

where $tf(ts, w)$ denotes the frequency of w in the text segment ts, N^d is the total number of text segments and text descriptions of images in web document d, and $tsf^d(w)$ is the text segment frequency of term w in web document d.

2.2 Visual Feature Extraction from Images

For an image downloaded, we first segment it into 10×10 rectangle regions. For each region we extract a visual feature vector, consisting of 6 color features and 60 gabor texture features which have been proven to be useful in many applications. Color features are the means and variances of the RGB color spaces. Texture features are extracted by calculating the means and variations of the Gabor filtered image regions on 6 orientations at 5 scales (frequencies). After the visual feature vectors of the image regions are extracted, all image regions are clustered using the k-means algorithm with k=500. The generated clusters, called *visterms*, are treated as a vocabulary for the images. For enriching this vocabulary of visterms with the high-level semantic features, a face detection model is used for detecting the faces in the images, which we found useful for understanding the contents of images in the domain of terror attack. Finally, a vector of visterm frequencies (501 dimensions) is extracted for each image.

3 Identifying Associations Between Texts and Images

3.1 Similarity-Based Retrieval Model

The task of identifying image-text associations can be cast into an *information retrieval* (*IR*) problem. Within a web document d containing images and text

segments, we treat each image i in d as a query to find a text segment ts that is most *semantically related* to i. Suppose each image i is represented by a visterm vector, denoted as i_v, together with a term vector of its surrounding text, denoted as i_t. For calculating the similarity between the images and text segments, we need to define a similarity measure $sim_d(i, ts) = sim_d(< i_v, i_t >, ts)$.

For simplifying the problem, we assume that, once an image i and a text segment ts are given, the similarity between i_v and ts and the similarity between i_t and ts are independent. Therefore, we can calculate $sim_d(i, ts)$ with the use of a linear mixture model as follows:

$$sim_d(i, ts) = sim_d(< i_v, i_t >, ts) = \lambda \cdot sim_d^{tt}(i_t, ts) + (1 - \lambda) \cdot sim_d^{vt}(i_v, ts). \quad (2)$$

3.2 Text-Based Similarity Measure

We use cosine distance as in Eq. 3 to measure the similarity between the textual features of an image and a text segment. The cosine distance measure is used as it has been proven to be insensitive to the length of text documents.

$$sim_d^{tt}(i_t, ts) = \frac{\sum_{k=1}^{n} w_k^d(i_t) w_k^d(ts)}{\| i_t \| \| ts \|} \quad (3)$$

3.3 Cross-Media Similarity Measure

Measuring similarity between visual and textual features is similar to the task of measuring relevance of documents in the field of multilingual retrieval for selecting documents in one language based on queries expressed in another. For multilingual retrieval, transformations are usually needed for bridging the gap between different representation schemes based on different terminologies. An open problem is that there is usually a basic distinction between the vocabularies of different languages, i.e. word senses may not be organized with words in the same way in different languages. Therefore, an exact mapping from one language to another language may not exist. This problem can be more serious in visual-textual transformation. Individual visterms can hardly convey any meaningful semantics without considering the contexts where they are placed. However, words in natural languages usually have relatively complete meanings. We can imagine that in most cases visterms can hardly be directly and precisely mapped to words because of the ambiguity. Vague transformations [2] [4] which have been proved useful in IR seem suitable for solving the vague problem of mapping visual representations of images to textual representations of text segments. In this paper, we borrow the idea from statistical vague transformation methods in multilingual retrieval for our cross-media similarity measure.

A drawback of the existing methods [2] [4] is that they require a large training set to build multilingual thesauruses. Such a training set is usually unavailable. In addition, as the construction of the multilingual thesauruses requires calculating an association factor for each pair of words picked from two languages, it may be computationally formidable. To overcome these obstacles, we introduce

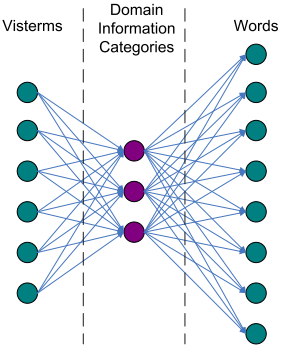


Fig. 2. An illustration of cross-media transformation with information bottleneck

an intermediate layer for the transformation. This intermediate layer is a set of domain information categories which can be seen as another vocabulary of a smaller size for describing domain information. For example, in terror attack domain, information categories may include *Attack Details*, *Impacts*, and *Victims* etc. Therefore, our cross-media transformation is in fact a concatenation of two sub-transformations, i.e. from visterm space to domain information categories and then to word space (see Figure 2). This is actually known as the *information bottleneck* method. For each sub-transformation, as the number of domain information categories is small, the size of the training data set for thesaurus construction needs not be large and the construction cost can be affordable. As discussed in Section 1, for an associated pair of image and text, their contents may not be an exact match or mapping. However, we believe that they can always be matched on a general domain information category.

Based on the above discussion, we aim to build two thesauri in the form of transformation matrices, each of which corresponds to a sub-transformation. Suppose the visterm space $\mathcal{V}$ has m dimensions and the textual feature space $\mathcal{T}$ has n dimensions. In addition, we suppose the cardinality of the set of high-level domain information categories $\mathcal{C}$ is l. Based on $\mathcal{V}$, $\mathcal{T}$, and $\mathcal{C}$, we define the two following transformation matrices:

$$M^{\mathcal{VC}}_{m\times l} = \begin{pmatrix} m^{\mathcal{VC}}_{11} & m^{\mathcal{VC}}_{12} & . & . & m^{\mathcal{VC}}_{1l} \\ m^{\mathcal{VC}}_{21} & m^{\mathcal{VC}}_{22} & . & . & m^{\mathcal{VC}}_{2l} \\ . & . & & & . \\ m^{\mathcal{VC}}_{m1} & m^{\mathcal{VC}}_{m2} & . & . & m^{\mathcal{VC}}_{ml} \end{pmatrix}, \quad M^{\mathcal{CT}}_{l\times n} = \begin{pmatrix} m^{\mathcal{CT}}_{11} & m^{\mathcal{CT}}_{12} & . & . & . & m^{\mathcal{CT}}_{1n} \\ m^{\mathcal{CT}}_{21} & m^{\mathcal{CT}}_{22} & . & . & . & m^{\mathcal{CT}}_{2n} \\ . & . & & & & . \\ m^{\mathcal{CT}}_{l1} & m^{\mathcal{CT}}_{l2} & . & . & . & m^{\mathcal{CT}}_{ln} \end{pmatrix}; \tag{4}$$

where $m^{\mathcal{VC}}_{ij}$ represents the association factor between the visterm v_i and the information category c_j; and $m^{\mathcal{CT}}_{jk}$ represents the association factor between the information category c_j and the textual feature t_k. Currently, $m^{\mathcal{VC}}_{ij}$ and $m^{\mathcal{CT}}_{jk}$ are calculated by

$$m^{\mathcal{VC}}_{ij} = P(c_j|v_i) \approx \frac{\#(v_i, c_j)}{\#(v_i)}, \quad m^{\mathcal{CT}}_{jk} = P(t_k|c_j) \approx \frac{\#(c_j, t_k)}{\#(c_j)}; \tag{5}$$

where $\#(v_i)$ is the number of images containing the visterm v_i; $\#(v_i, c_j)$ is the number of images containing v_i and belonging to the information category c_j; $\#(c_j)$ is the number of text segments belonging to the category c_j; and $\#(c_j, t_k)$ is the number of text segments belonging to c_j and containing the term t_k.

Based on Eq. 4, we can define the similarity between the visual part of an image i_v and a text segment ts as $i_v^T M_{m\times l}^{\mathcal{VC}} M_{l\times n}^{\mathcal{CT}} ts$. For embedding into Eq. 2, we use its normalized form

$$sim^{\mathcal{VT}}(i_v, ts) = \frac{i_v^T M_{m\times l}^{\mathcal{VC}} M_{l\times n}^{\mathcal{CT}} ts}{\parallel i_v^T M_{m\times l}^{\mathcal{VC}} M_{l\times n}^{\mathcal{CT}} \parallel \parallel ts \parallel}. \tag{6}$$

Eq. 6 calculates the cross-media similarity using a *single-direction transformation* from visterm space to word space. However, it may still cause vague problems. For example, suppose there is a picture i belonging to a domain information category, *Attack Details*, and two text segments $ts1$ and $ts2$ belonging to the categories of *Attack Details* and *Victims* respectively. If the two categories, *Attack Details* and *Victims*, share many common words (such as *kill*, *die*, and *injure*), the transformation result of i_v might be similar to both $ts1$ and $ts2$. To reduce the influence of common terms in different categories and employ the strength of the distinct words, we consider another transformation from word space to visterm space. We can similarly define another pair of transformation matrices $M_{n\times l}^{\mathcal{TC}} = \{m_{kj}^{\mathcal{TC}}\}^{n\times l}$ and $M_{l\times m}^{\mathcal{CV}} = \{m_{ji}^{\mathcal{CV}}\}^{l\times m}$, where $i = 1, 2, ..., m$, $j = 1, 2, ..., l$, and $k = 1, 2, ..., n$. Then, the similarity from a text segment ts to the visual part of an image i_v can be defined as

$$sim^{\mathcal{TV}}(ts, i_v) = \frac{ts^T M_{n\times l}^{\mathcal{TC}} M_{l\times m}^{\mathcal{CV}} i_v}{\parallel ts^T M_{n\times l}^{\mathcal{TC}} M_{l\times m}^{\mathcal{CV}} \parallel \parallel i_v \parallel}. \tag{7}$$

Finally, we can define a cross-media similarity measure using the *dual-direction transformation* which is the geometric mean of $sim^{\mathcal{VT}}(i_v, ts)$ and $sim^{\mathcal{TV}}(ts, i_v)$:

$$sim_d^{vt}(i_v, ts) = \sqrt{sim^{\mathcal{VT}}(i_v, ts) \cdot sim^{\mathcal{TV}}(ts, i_v)}. \tag{8}$$

4 Experiments

The experiments are conducted on an image collection, containing 285 images related to terrorist attacks, downloaded from the CNN and BBC news web sites. We manually categorize about 1500 text segments and 285 images into twelve domain information categories, i.e. *Anti-Terror, Attack Details, After Attack, Government Responses, Rescure, Impact, Investigation, Terrorist Claims, Terrorist Suspects, Victims, Ceremony*, and *Others*. We use a 5-fold cross-validation to test the performance of our method in terms of precision defined by $precision = \frac{\#(Correctly\ Identified\ Associations)}{\#(Total\ Images)}$. The correctness of the extracted image-text associations are judged by human by inspecting the web pages wherein the images appear.

Caption In Web Pages	Police photograph the body of the gunman.	Wreckage of the base of the World Trade Center. The CIA searched the wreckage.	Injured man being helped away.
Cross-Media Measure ($\lambda = 0.0$)	**At least five people have died, and several others have been injured, in several incidents, including a shooting by a Palestinian gunman in the Israeli town of Kfar Saba, and a suicide bomb attack in north Jerusalem. (SC=0.129)**	A secret CIA office was destroyed in the 11 September attack on the World Trade Center, the New York Times reports. (SC=0.142)	**It was here on Thursday that a Palestinian suicide bomber blew himself up on board a crowded bus, killing five people and injuring about 50 others. (SC=0.085)**
Text-Based Measure ($\lambda = 1.0$)	**At least five people have died, and several others have been injured, in several incidents, including a shooting by a Palestinian gunman in the Israeli town of Kfar Saba, and a suicide bomb attack in north Jerusalem. (SC=0.089)**	**The CIA sent a special team to scour the wreckage for vital intelligence reports after the attack, the paper says. (SC=0.268)**	Others were not even able to do that. One witness said he saw several people lying on the floor of the bus, including one man whose legs had been blown off. (SC=0.110)
Mixture Measure ($\lambda = 0.6$)	**At least five people have died, and several others have been injured, in several incidents, including a shooting by a Palestinian gunman in the Israeli town of Kfar Saba, and a suicide bomb attack in north Jerusalem. (SC=0.104)**	**The CIA sent a special team to scour the wreckage for vital intelligence reports after the attack, the paper says. (SC=0.216)**	Others were not even able to do that. One witness said he saw several people lying on the floor of the bus, including one man whose legs had been blown off. (SC=0.084)

Fig. 3. A sample set of image-text associations extracted with similarity scores (SC). The correctly identified associated texts are bolded.

As indicated by the experimental results shown in Table 1, we see that textual information is essential for identifying image-text associations. In fact, pure text similarity measure ($\lambda = 1.0$) outperforms pure cross-media similarity measure ($\lambda = 0.0$) by 23.0%-25.7% in terms of average precision. However, the best result (average precision of 67.2%) is achieved by the linear mixture model using both text-based and cross-media similarity measures (with $\lambda = 0.6$). This shows that visual features are useful for improving the performance of the identification task. In fact, we observe that keywords extracted from surrounding texts of images sometimes may be inconsistent with the contents of the images. Visual features can provide more information for the disambiguation of the image semantics and reducing the influence of the imprecision caused by textual features. Another important observation is that pure dual-direction transformation is better than pure single-direction transformation for measuring the cross-media similarity ($\lambda = 0.0$). In general, the overall precision of using dual-direction transformation is higher than that of using single-direction transformation.

A sample set of the extracted image-text associations is shown in Figure 3. We notice that when text contents in web pages are quite different, e.g. belonging

Table 1. The precision scores (%) for image-text association extraction

Fold	Transformation Used	λ										
		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1	Single-Direction	41.5	56.6	62.3	69.8	**71.7**	69.8	66.0	66.0	66.0	66.0	62.3
	Dual-Direction	43.4	56.6	64.2	67.9	**73.6**	69.8	69.8	69.8	67.9	67.9	62.3
2	Single-Direction	32.1	45.3	47.2	58.5	60.4	60.4	**64.2**	**64.2**	**64.2**	62.3	62.3
	Dual-Direction	37.7	45.3	47.2	60.4	64.2	62.3	**66.0**	64.2	64.2	62.3	62.3
3	Single-Direction	26.4	34.0	45.3	54.7	62.3	64.2	**69.8**	**69.8**	**69.8**	67.9	66.0
	Dual-Direction	26.4	34.0	43.4	54.7	62.3	66.0	**69.8**	**69.8**	**69.8**	67.9	66.0
4	Single-Direction	37.7	49.1	56.6	62.3	64.2	64.2	66.0	**67.9**	66.0	64.2	64.2
	Dual-Direction	41.5	47.2	56.6	66.0	64.2	66.0	**67.9**	**67.9**	**67.9**	64.2	64.2
5	Single-Direction	43.4	58.5	66.0	64.2	64.2	**66.0**	64.2	62.3	58.5	54.7	54.7
	Dual-Direction	45.3	54.7	60.4	64.2	**66.0**	64.2	62.3	62.3	60.4	54.7	54.7
Average	Single-Direction	36.2	48.7	55.5	61.9	64.5	64.9	**66.0**	**66.0**	64.9	63.0	61.9
	Dual-Direction	38.9	47.5	54.3	62.6	66.0	65.7	**67.2**	66.8	66.0	63.4	61.9

to different domain information categories, the cross-media similarity measure may be efficient enough to identify the associated text for an image (see the first and third column in Figure 3). For the case that contents in different text segments are related to each other, using only the cross-media similarity measure may not identify the most suitable text segment, but the extracted one can be semantically relevant (the second column in Figure 3).

5 Conclusion

In this paper, we present an approach for extracting associations between images and texts from web pages for cross-media information fusion. We use a similarity-based multilingual retrieval model and adopt a vague transformation technique for measuring the similarity between visual features and textual features. The experimental results suggest that combination of visual and textual features can produce better results than using visual or textual features alone.

References

1. Radev, D.R.: A common theory of information fusion from multiple text sources step one: cross-document structure. In: Proceedings of the 1st SIGdial workshop on Discourse and dialogue, Morristown, NJ, USA, Association for Computational Linguistics (2000) 74–83
2. Mandl, T.: Vague transformations in information retrieval. In: ISI. (1998) 312–328
3. Chang, S.F., Manmatha, R., Chua, T.S.: Combining text and audio-visual features in video indexing. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005 (ICASSP '05). (2005) 1005–1008
4. Sheridan, P., Ballerini, J.P.: Experiments in multilingual information retrieval using the spider system. In: SIGIR '96, New York, ACM Press (1996) 58–65

Mining Sequences of Temporal Intervals

Steffen Kempe and Jochen Hipp

DaimlerChrysler AG, Group Research, 89081 Ulm, Germany
{Steffen.Kempe, Jochen.Hipp}@daimlerchrysler.com

Abstract. Recently a new type of data source came into the focus of knowledge discovery from temporal data: interval sequences. In contrast to event sequences, interval sequences contain labeled events with a temporal extension. However, existing algorithms for mining patterns from interval sequences proved to be far from satisfying our needs. In brief, we missed an approach that at the same time: defines support as the number of pattern instances, allows input data that consists of more than one sequence, implements time constraints on a pattern instance, and counts multiple instances of a pattern within one interval sequence. In this paper we propose a new support definition which incorporates these properties. We also describe an algorithm that employs the new support definition and demonstrate its performance on field data from the automotive business.

1 Introduction

Mining sequences from temporal data is a well known data mining task which gained a lot of attention in the past, e.g. [2,5]. In all these approaches, the temporal data is considered to consist of events. Each event has a label and a timestamp. In the following, we want to focus on temporal data where an event has a temporal extension. These temporally extended events are called temporal intervals. Each temporal interval can be described by a triplet (b, e, l) where b and e denote the beginning and the end of the interval and l its label. For example a sequence of temporal intervals may describe the medical history of a patient in a hospital or the data collected by a flight recorder.

At DaimlerChrysler we are interested in mining interval sequences in order to further extend the knowledge about our products. Thus, in our domain one interval sequence may describe the history of one vehicle. The configuration of a vehicle, e.g. whether it is an estate car or a limousine, can be described by temporal intervals. The build date is the begin and the current day is the end of such a temporal interval. Other temporal intervals may describe stopovers in a garage or the installation of additional equipment. Mining these interval sequences helps us in tasks like quality monitoring or improving customer satisfaction.

In the following section we give formal definitions of the mining task. In Section 3 we introduce a new support definition which is motivated by our experiences in the automotive industry. Next, we propose a novel algorithm for finding frequent patterns which implements the new support definition. The algorithm is tested on real world data from our domain in Section 5.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 569–576, 2006.
© Springer-Verlag Berlin Heidelberg 2006

2 Foundations

As previously mentioned, we represent a temporal interval as a triplet (b, e, l).

Definition 1. *(Temporal Interval) Given a set of labels $l \in L$, we say the triplet $(b, e, l) \in \mathbb{R} \times \mathbb{R} \times L$ is a temporal interval, if $b \leq e$. The set of all temporal intervals over L is denoted by I.*

Definition 2. *(Interval Sequence) Given a sequence of temporal intervals, we say $(b_1, e_1, l_1), (b_2, e_2, l_2), \ldots, (b_n, e_n, l_n) \in I$ is an interval sequence, if*

$$\forall (b_i, e_i, l_i), (b_j, e_j, l_j) \in I, i \neq j : b_i \leq b_j \wedge e_i \geq b_j \rightarrow l_i \neq l_j \tag{1}$$

$$\begin{gathered} \forall (b_i, e_i, l_i), (b_j, e_j, l_j) \in I, i < j : \\ (b_i < b_j) \vee (b_i = b_j \wedge e_i < e_j) \vee (b_i = b_j \wedge e_i = e_j \wedge l_i < l_j) \end{gathered} \tag{2}$$

hold. A given set of interval sequences is denoted by $\mathbb{S}$.

Equation 1 above is referred to as the *maximality assumption* [4]. The maximality assumption guarantees that each temporal interval A is maximal, in the sense that there is no other temporal interval in the sequence sharing a time with A and carrying the same label. Equation 2 requires that an interval sequence has to be ordered by the beginning (primary), end (secondary) and label (tertiary, lexicographically) of its temporal intervals.

Without temporal extension there are only two possible relations. One event is before (or after as the inverse relation) the other or they coincide. In case of temporal intervals there are 7 possible relations (respectively 13 including inverse relations). These interval relations have been described by Allen in [3] and are depicted in Figure 1. Each relation of Figure 1 is a temporal pattern on its own that consists of two temporal intervals. Patterns with more than two temporal intervals are straightforward. One just needs to know which interval relation exists between each pair of labels. Using the set of Allen's interval relations $\mathbb{I}$, a temporal pattern is defined by:

Definition 3. *(Temporal Pattern) A pair $P = (s, R)$, where $s : 1, \ldots, n \rightarrow L$ and $R \in \mathbb{I}^{n \times n}$, $n \in \mathbb{N}$, is called a "temporal pattern of size n".*

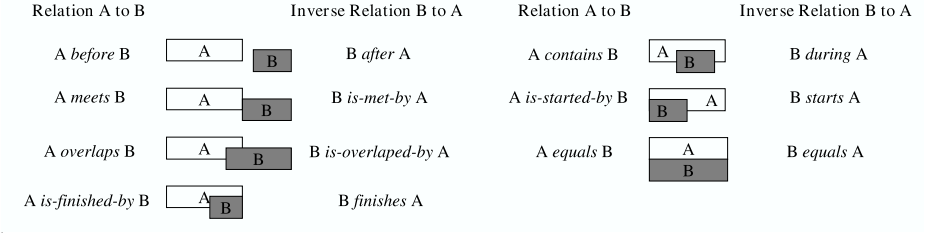
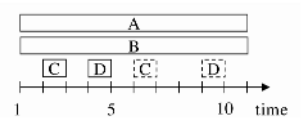


Fig. 1. Allen's Interval Relations

a)



b)

	A	B	C	D
A	e	***e***	***c***	***c***
B	e	e	***c***	***c***
C	d	d	e	***b***
D	d	d	a	e

Fig. 2. a) Example of an Interval Sequence b) Example of a Temporal Pattern (e is the abbreviation for *equals*, c for *contains*, etc.)

If a temporal pattern holds true (is valid) for an interval sequence we consider this sequence as an instance of the pattern.

Definition 4. *(Instance) A temporal pattern* $P = (s, R)$ *holds true for an interval sequence* $S = (b_i, e_i, l_i)_{1 \leq i \leq n}$*, if* $\forall i, j : s(i) = l_i \wedge s(j) = l_j \wedge R[i, j] = \text{ir}([b_i, e_i], [b_j, e_j])$ *with function* ir *returning the relation between two given intervals. We say that the interval sequence* S *is an instance of temporal pattern* P*. We say that an interval sequence* S' contains *an instance of* P *if* $S \subseteq S'$*, i.e.* S *is a subsequence of* S'*.*

Obviously a temporal pattern can only be valid if its labels have the same order as their corresponding temporal intervals have in an instance of the pattern.

Figure 2a) shows an example of an interval sequence which contains an instance of the temporal pattern given in Figure 2b).

The mining task is to find all temporal patterns in a given set of interval sequences which satisfy a user specified minimum support threshold. Note that this mining task is closely related to frequent itemset mining [1].

3 A New Support Definition

Previous investigations on discovering patterns from sequences of temporal intervals include the work of [4] and Papapetrou et al. [6]. Both approach the problem very differently from each other.

Höppner mines patterns from one sequence of temporal intervals only. The problem of multiple sequences is not addressed. He also requires that a pattern occurs within a certain time window. In contrast, Papapetrou et al. analyze multiple sequences but they do not have any time constraints on pattern instances.

The main difference between both approaches is the definition of support for a pattern. Höppner defines the *temporal support* of a pattern. This definition is closely related to Mannila's *frequency* in [5]. The temporal support can be interpreted as the probability to see an instance of the pattern within the time window if the time window is randomly placed on the temporal interval sequence. On the other hand, Papapetrou et al. count the number of instances for each pattern. The pattern counter is incremented once for each sequence that contains the pattern. If a temporal interval sequence contains multiple instances of a pattern then these additional instances will not further increment the counter.

Consider the pattern *C before D* in the example of Figure 2a). As the interval sequence contains an instance of the pattern the support of Papapetrou et al. is

1. Using a time window of 3 Höppner will calculate a support of 3 (a duration of 2 for the first occurrence and 1 for the second).

Yet, for our needs and comparable applications the accurate number of pattern instances is indispensable for generating knowledge in our domain. Thus we developed a new support definition which: 1. counts the number of pattern instances, 2. handles multiple instances of a pattern within one interval sequence, and 3. allows time constraints on a pattern instance.

In [5], Mannila et al. introduced *minimal occurrences* as a support definition for patterns in a single sequence of events. We extend the approach of minimal occurrences to the circumstances of temporal intervals.

Given a temporal pattern P and an interval sequence S, a time window $[t_b, t_e]$ of S is called *minimal occurrence* of P, if there is an instance of P in the time window but not in one of its proper subwindows.

A special case is if the temporal pattern P is of size 1, i.e. it contains only one label. Then every temporal interval (b, e, l) with $b < e$ leads to an infinite number of minimal occurrences. Therefore the minimal occurrences of P are $[b, e]$ for each temporal interval (b, e, l) and l is the label of P.

Minimal occurrences also provide an easy way to introduce time constraints on a pattern instance. Suppose a pattern instance is only valid if it occurs within a certain period of time, then we just need to count those minimal occurrences whose lengths do not exceed the time limit.

After we introduced minimal occurrences in our application, we realized that the support definition is still not sufficient. There is a subset of temporal patterns whose supports are calculated differently than we expected. Figure 2a) gives an example of such a temporal pattern (solid lines). The minimal occurrence is $[1, 11]$. In any smaller time window, the relation *equals* between the temporal intervals A and B is not visible. Thus if an interval sequence contains a second C *before* D during the temporal intervals A and B (dashed lines in Figure 2a)), it will not be counted as it produces the same minimal occurrence. This example is important to us because the temporal intervals C and D might describe garage stopovers for a car with A and B specifying its vehicle configuration. So we want to count multiple instances of temporal patterns (here C *before* D) for different combinations of configurations. Hence, in the example above, we have to count the minimal occurrences of the subpattern C *before* D given that the subpattern is contained in temporal intervals A and B.

As a result, we have to distinguish those temporal patterns from all other patterns where a subpattern is decisive for the support calculation. In the latter patterns there exists a subpattern which is contained in all other temporal intervals of the temporal pattern. We can decide whether a given temporal pattern P contains such a subpattern by transforming the problem into graph theory based on the upper triangular matrix of its relation table. We start by creating an empty graph. For each label of the temporal pattern we insert a vertex into the graph. Next we create an edge for each relation in the upper triangular matrix of the relation table which is not *contains* between the corresponding vertices in the graph. If the resulting graph is unconnected then there is a subpattern

which is contained in all other temporal intervals of the pattern. This subpattern corresponds to one of the connected subgraphs.

We call a temporal pattern connected if its graph is *connected*. Otherwise we call the temporal pattern *unconnected*. In contrast to [5] we define the support of a connected pattern P as its total number of minimal occurrences in all sequences of $\mathbb{S}$. If P is unconnected the support is given by the total number of minimal occurrences of its subpattern.

4 New Algorithm

As in existing approaches, the main idea is to generate all frequent temporal patterns by applying the Apriori scheme of candidate generation and support evaluation. These two steps are alternately repeated until no more candidates are generated. Apriori starts with the frequent 1-patterns and then successively derives all k-candidates from the set of all frequent (k-1)-patterns. This approach requires that each subpattern of a frequent pattern is frequent. Unfortunately the extension of minimal occurrences to temporal intervals destroys this downward closure property. The problem arises if a temporal interval is used as the same part of a pattern for multiple instances. Consider, e.g., the interval sequence (1,11,A), (2,3,C), (6,7,C) (see Figure 2a)). There are two minimal occurrences of the pattern *A contains C* $[2, 3]$ and $[6, 7]$ but there is only one minimal occurrence of *A* $[1, 11]$. Here *A* is used twice as the same part of the pattern. Hence, the downward closure property is not guaranteed for minimal occurrences.

A closer investigation shows that *contains* is the only relation for which the property does not hold. Consider the pattern *A overlaps B*. The downward closure property can only fail if *A* overlaps two or more *B*'s. This is impossible as these *B*'s would have to share a time interval which is prohibited by the maximality assumption (Equation 1). The same argument holds for *meets*, *is-finished-by*, *is-started-by*, *equals* and their inverse relations. In case of *A before B* there can be several *B*'s after the *A* but the definition of minimal occurrences allows the first *B* to be counted.

Obviously the downward closure property only fails for unconnected temporal patterns. Our approach to solve this problem is by treating connected and unconnected temporal patterns differently.

When generating candidates in the Apriori-way, the candidate pattern *A equals B, A and B contain C* would be generated based on the two subpatterns *A equals B* and *A contains C*. However, *A equals B* needs not necessarily be frequent even if the resulting candidate *A equals B, A and B contain C* is frequent.

In the example above, the subpatterns *A contains C* and *B contains C* are necessarily frequent if the candidate itself is frequent. Generally, in a valid unconnected (k+1)-pattern, $k \geq 2$, there exists a $j, 1 \leq j \leq k$, such that the first j labels always describe those temporal intervals which contain all other temporal intervals of the pattern (labels $j + 1, \ldots, k + 1$). *Contains* implies "starts before

and ends after", so the ordering of sequences guarantees exactly this property. Hence, in the opposite way, the last $k+1-j$ labels of the pattern are always responsible for the frequency of the pattern. If we remove the first or second label from a frequent (k+1)-pattern the resulting k-patterns must still be frequent. In other words, generating (k+1)-candidates by joining the two subpatterns that share the last k–1 labels ensures that the set of generated (k+1)-candidates is a superset of the frequent (k+1)-patterns. We only need to guarantee that the initial set of frequent 2-patterns contains all frequent unconnected patterns.

Thus, by modifying the candidate generation step, we can transfer the completeness property of the Apriori approach, to unconnected temporal patterns. In detail, our approach of candidate generation is as follows: The candidate patterns of size 1 are generated by using all available labels in the dataset. In the next step, we use all 1-candidates to create the candidate patterns of size 2. This guarantees that we will find all frequent 2-patterns, albeit they are connected or unconnected. For the actual candidate generation and test approach the frequent patterns of size k ($k \geq 2$) are used to generate the candidate patterns of size k+1. This is achieved by joining every pair of temporal patterns P and Q which are identical w.r.t. the last k-1 rows and columns of their relation table, i.e. they share common (k–1)-pattern on the last k-1 labels. Each temporal pattern describes the desired (k+1)-pattern except for one label. If we join P and Q to the temporal pattern R, as it is illustrated in Figure 3, then there is only one interval relation missing in R.

The missing interval relation describes the relation between the first labels of P and Q (l_1^p and l_1^q). Now we can extend R to a set of candidates by applying Allen's interval relations. The missing value in R is substituted by each of Allen's interval relations. Hence, the extension of R leads to 7 (k+1)-candidate patterns.

The second step is the support evaluation of the candidate patterns. As already mentioned, the labels of a valid temporal pattern have the same order as their counterparts in an instance would have. Therefore we can find an instance of a temporal pattern by using finite state machines which subsequently take the temporal intervals of an ordered temporal sequence as input.

It is straightforward to derive a finite state machine from a temporal pattern. For each label in the pattern a state is generated. The finite state machine starts in an initial state. The next state is reached if we input a temporal interval that contains the same label as the first label of the temporal pattern. From now on the next states can only be reached if the shown temporal interval carries

Pattern P	l_1^p	$l_2 \ldots l_k$
l_1^p	e	C
l_2		
$\vdots$	B	A
l_k		

$\cup$

Pattern Q	l_1^q	$l_2 \ldots l_k$
l_1^q	e	E
l_2		
$\vdots$	D	A
l_k		

$\Longrightarrow$

Pattern R	l_1^p	l_1^q	$l_2 \ldots l_k$
l_1^p	e	?	C
l_1^q	?	e	E
l_2			
$\vdots$	B	D	A
l_k			

Fig. 3. Temporal Patterns P, Q share a k-1 subpattern, joining P and Q yields R

the same label as the state and its interval relation to all previously accepted temporal intervals is the same as specified in the temporal pattern. If the finite state machine reaches its last state it also reaches its final accepting state.

We can derive the minimal time window in which this particular pattern instance is visible from the set of temporal intervals which has been accepted by the state machine. We know that the time window contains an instance of the pattern but we do not know whether it is a minimal occurrence. Therefore we apply a two step approach. First we will find all occurrences of a pattern using state machines. Second we will filter out all occurrences which are not minimal.

To find all occurrences of a pattern in an interval sequence we are maintaining a set of finite state machines. At first, the set only contains the finite state machine that is derived from the candidate pattern. Subsequently, each temporal interval from the interval sequence is shown to every finite state machine in the set. If a finite state machine can accept the temporal interval a copy of the state machine is made and added to the set. The temporal interval is shown to only one of these two state machines . Hence, there will always be a copy of the initial state machine in the set trying to find an occurrence of the pattern.

5 Performance Evaluation and Conclusions

In order to evaluate the performance of the algorithm we employed a dataset from our domain. This dataset contains information about 101 250 vehicles. We performed 5 different experiments varying the minimum support threshold from 3 200 down to 200.

Figure 4a) shows the number of candidates that are generated in each iteration. Obviously the number of candidates grows rapidly as the minimum support threshold gets lower. This general behaviour is well known from frequent itemset mining. In contrast to frequent itemsets, Figure 4a) shows two distinct peaks. There is one peak for the candidate patterns of size 2 and one peak for patterns of sizes 6-7. Moreover, the first peak does not vary with different minimum support thresholds. This peak is a result of the special candidate generation in the first iteration of our algorithm. The candidates of size 2 are generated by using all the candidates of size

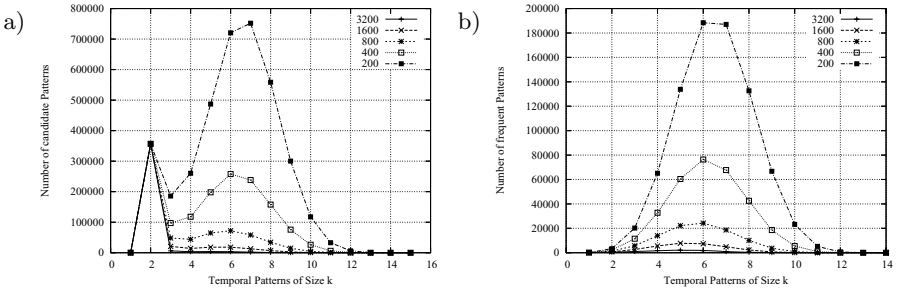


Fig. 4. a) Candidate Patterns generated b) Frequent Patterns found in each Iteration

1 (the frequent patterns are only used in subsequent iterations). Consequently the number of 2-candidates is independent of the chosen minimum support threshold.

In Figure 4b) the number of frequent patterns is depicted for each iteration. For each minimum support threshold the maximum number of frequent patterns is found between the 5-th and 7-th iteration. Again the number of frequent patterns grows rapidly with decreasing minimum support thresholds.

The increasing number of frequent and candidate patterns also leads to longer runtimes of the algorithm for decreasing minimum support thresholds. While the first experiment (MinSupp=3 200) was finished within 36 minutes subsequent experiments took 98, 295, 1052 and 2822 minutes.

In this paper we presented a new algorithm for discovering frequent temporal patterns. The key advantages of this algorithm are the ability to mine data that consists of several separate interval sequences, a new support definition that allows counting multiple instances of a pattern per sequence, and finally the consideration of time constraints on pattern instances.

Whereas on its own, these features have been described before, e.g. [4,6], an algorithm implementing them all together had not yet been available. For our and many other applications combining these features is essential. Thus, our approach opens a broad range of new applications for sequence mining.

Combining the sketched features is far from being straightforward. The main algorithmic challenge is that the downward closure property of support, also known as the Apriori-criteria is not met. In other words, we had to develop an algorithm that is efficient and still complete with respect to a minimal support threshold, although subpatterns of a frequent pattern may be infrequent. We finally tackled the issue by distinguishing so called connected and unconnected patterns based on the temporal relation *contains*.

Based on an analysis of warranty data from the automotive domain, we showed that the algorithm can be successfully applied to real world data. The results contained valuable knowledge far beyond current approaches and were produced within reasonable time.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int. Conf. on Very Large Databases (VLDB '94)*, pages 487–499, 1994.
2. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the 11th Int. Conf. on Data Engineering (ICDE '95)*, pages 3–14, 1995.
3. J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.
4. F. Höppner and F. Klawonn. Finding informative rules in interval sequences. *Intelligent Data Analysis*, 6(3):237–255, 2002.
5. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
6. P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Discovering frequent arrangements of temporal intervals. In *5th IEEE Int. Conf. on Data Mining*, 2005.

Pattern Teams

Arno J. Knobbe[1,2] and Eric K.Y. Ho[1]

[1] Kiminkii, Postbus 171, NL-3990 DD, Houten, The Netherlands
a.knobbe@kiminkii.com, e.ho@kiminkii.com
[2] Utrecht University, P.O. box 80 089, NL-3508 TB Utrecht, The Netherlands

Abstract. Pattern discovery algorithms typically produce many interesting patterns. In most cases, patterns are reported based on their individual merits, and little attention is given to the interestingness of a pattern in the context of other patterns reported. In this paper, we propose filtering the returned set of patterns based on a number of quality measures for pattern sets. We refer to a small subset of patterns that optimises such a measure as a *pattern team*. A number of quality measures, both supervised and unsupervised, is proposed. We analyse to what extent each of the measures captures a number of 'intuitions' users may have concerning effective and informative pattern teams. Such intuitions involve qualities such as independence of patterns, low overlap, and combined predictiveness.

1 Introduction

Over the last few years, there has been an increasing interest in pattern discovery algorithms. In this branch of Data Mining, the emphasis lies on discovering a collection of local patterns that satisfy a number of inductive constraints provided by the user, rather than on the induction of a single global model of the data. Typically, a pattern represents some subgroup of the data, and patterns are selected on the basis of the support of the subgroup and one or more constraints on interestingness measures, for example based on the correlation with a target concept. Common examples of such Data Mining settings are frequent pattern discovery (itemsets, trees, graphs, etc.) [11], association rule mining [12] and subgroup discovery [1, 3, 9, 10, 12]. In most cases, patterns are reported based on their individual merits, and little attention is given to the interestingness of a pattern in the context of other patterns reported. As a result, the outcome of a pattern discovery exercise is often a large collection of patterns, with high levels of redundancy, that is hard to inspect manually. It is our aim in this paper to improve the effectiveness of pattern discovery algorithms by considering the quality of patterns in the context of other patterns reported.

Let us consider a busy end-user, who has only very limited time to inspect the outcome of a pattern discovery exercise. If only a few patterns can be considered, a small yet effective set of patterns needs to be selected. Having already seen one or more patterns, the next pattern presented needs to both perform well, and be substantially different from the first patterns. If the next pattern effectively covers almost the same set of individuals as any combination of the previous ones (even though syntactically it might be completely different), it provides little new information. We present a method of selecting a small subset of patterns – a *pattern team* – that optimises some

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 577 – 584, 2006.
© Springer-Verlag Berlin Heidelberg 2006

given quality measure for sets of patterns. In this paper we consider four candidate quality measures that promote different desirable properties of sets of patterns. Depending on the (implicit) expectations of the end-user, different methods of pattern selection can thus be applied, resulting in different pattern teams.

The quality measures presented are inspired by a number of intuitions end-users typically have about pattern mining results. We use the following list of intuitions, and test to what extent the four measures satisfy these intuitions:

I1 No two patterns should cover (approximately) the same set of examples.
I2 No pattern should cover (approximately) the complement of another pattern.
I3 No pattern should cover (approximately) a logical combination of two or more other patterns.
I4 Patterns should be (approximately) mutually exclusive.
I5 When using patterns as input to a classifier, the pattern set should lead to the best performing classifier.
I6 Patterns should lie on the convex hull of all patterns in ROC-space.

Clearly, these six intuitions cannot all hold at the same time. In fact, some intuitions are to some extent competing (e.g. I2 and I4). One should think of these intuitions as descriptions of the kind of expectations an end-user may have about the set of patterns returned. Typically, we will only be interested in satisfying one, or a few of these intuitions.

An important characteristic of the four quality measures presented is that they are syntax-independent (although one can envisage more syntax-oriented measures). This means that pattern sets are solely judged on the subgroups of individuals covered by each pattern, and the potential overlap or independence of these subgroups. As a result, our methodology applies to any mining paradigm where patterns represent subgroups. This includes complex domains such as structured and multi-relational domains, where rich pattern languages make purely syntactical comparisons of patterns difficult or expensive.

An extended version of this paper can be found in [5]. This version contains more examples, experiments and extended descriptions of related work.

2 Pattern Team Discovery

The process of filtering patterns will generally be preceded by a pattern discovery phase. The *pattern discovery* task can be defined as follows: given a pattern collection $\boldsymbol{P}$, a set of interestingness measures $\phi_1, \ldots, \phi_l$, $\phi_i : \boldsymbol{P} \rightarrow [0, 1]$, and a set of threshold values $\sigma_1, \ldots, \sigma_l$, $\sigma_i \in [0, 1]$, find all patterns $p \in \boldsymbol{P}$ such that $\forall i : \phi_i(p) \geq \sigma_i$. In an alternative setting, the top k patterns with respect to one of the interestingness measures is returned.

In both settings, the outcome will typically be a large set of interesting patterns P_ϕ with considerable levels of redundancy. In this paper, we therefore propose a second phase, consisting of a *pattern team discovery* task: given a set of interesting patterns P_ϕ, and a quality measure for pattern sets $\Phi : 2^{\boldsymbol{P}} \rightarrow \mathbf{R}$, find a pattern set $P \subseteq P_\phi$ of size k such that $\Phi(P) \geq \Phi(Q)$ for all $Q \subseteq P_\phi$ of size k.

In this case, we are interested in a single pattern team of specified size k, but other variants of this task can be imagined. For example, one could query for *all* pattern teams of size k, or for an optimal pattern team regardless of its size. Alternatively, one can imagine a discovery task that returns all (or the top k) pattern sets P such that their quality exceeds some threshold: $\Phi(P) \geq \Sigma$. This setting however defies the purpose of pattern filtering, as potentially many pattern sets will be returned.

Finding a pattern team of size k for any given pattern set quality measure Φ potentially involves the consideration of $\binom{n}{k}$ subsets of P_ϕ, where $n = |P_\phi|$. In fact, Mielikäinen et al. [7] show that the general pattern team discovery problem is NP-hard by relating it to the set-covering problem, making it infeasible for all but small values of k. Fortunately, for specific quality measures, it is possible to find optimal pattern sets efficiently, or to find approximations that can be shown to perform reasonably well [7]. In [4], we provide some example algorithms for *joint entropy*, one of the quality measures considered in this paper. The thorough treatment of efficient implementations of pattern team discovery is outside the scope of this paper.

3 Quality Measures

In this section, we present four quality measures for pattern sets. Two of the four measures work in an unsupervised fashion: they consider properties of the subgroups covered (specifically independence and mutual exclusiveness), and ignore the potential predictive qualities of a pattern. Note that these two measures work on pattern sets discovered by algorithms working in either supervised or unsupervised fashion. Conversely, we could create a pattern team using one of the two supervised quality measures on patterns discovered in unsupervised mode (e.g. frequent patterns). We thus have a choice for supervised or unsupervised both for the initial pattern discovery phase, as well as the subsequent pattern filtering phase.

The following basic definitions will allow us to define the four quality measures for pattern sets formally. We assume that our database d is a bag of labelled objects $i \in D$, referred to as *individuals*, taken from a domain D. Furthermore there is a function l: $d \rightarrow \mathbf{R}$ that specifies the label of an individual. If the labels just have values 0 and 1, we interpret the individuals as belonging to the negative (F) and positive (T) classes, respectively. Alternatively, we treat the mining task as a regression problem. We refer to the size of the database as $N = |d|$.

We assume nothing about the syntax of the pattern language, and treat a pattern simply as a function p: $D \rightarrow \{0, 1\}$. We will say that a pattern p *covers* an individual i *iff* $p(i) = 1$. A *subgroup* $S(d, p)$ implied by a pattern is now simply the set of individuals $i \in d$ that are covered by p: $S(d, p) = \{i \in d \mid p(i) = 1\}$. For brevity we will omit the d from now on. $s(p) = |S(p)|$ refers to the size of the subgroup implied by p. Furthermore, we will use expressions like $l(i) = 1$ to denote patterns related to the label of individuals, such that $S(l(i) = 1)$ for example denotes the set of positive cases.

When talking about sets of patterns $P = \{p_1, \ldots, p_k\}$ of size k, an individual may be covered by some patterns in P and not by others. In order to represent such *contingencies*, we introduce *codes* $c \in \{0, 1\}^k$. The subgroup implied by a given pattern set

P and a code c is defined by $S(P, c) = \{i \in d \mid p_1(i) = c_1, \ldots, p_k(i) = c_k\}$. $s(P, c) = |S(P, c)|$ is the size of the subgroup implied by P and c.

Joint Entropy. The first quality measure for pattern teams is based on our work on maximally informative k-itemsets (miki's) [4]. In essence, we treat each pattern in P_ϕ as a binary feature (an item), and for each pattern set $P \subseteq P_\phi$ of size k, compute the joint entropy [4] of the features in P. A miki is then simply the itemset (pattern set) that optimises this joint entropy. Our first quality measure Joint Entropy $H(P)$ is hence defined as:

$$H(P) = -\sum_{c \in \{0,1\}^k} \frac{s(P,c)}{N} \lg \frac{s(P,c)}{N}$$

The entropy is a measure for the uniformity of the distribution of individuals over the different contingencies. A uniform distribution is achieved if for all patterns $s(p) = N / 2$, and all patterns are independent. A pattern team that optimises the joint entropy will hence optimise the power to distinguish between individuals. Note that H is unsupervised, as well as insensitive to replacing one or more patterns with their complement [4]. Patterns are merely used to distinguish two complementary sets of individuals.

Exclusive Coverage. The second quality measure is inspired by intuition I4: pattern sets that reduce the amount of overlap between patterns are favoured. Because overlap is less likely with patterns of low support, we will also have to promote the support of individual patterns. The Exclusive Coverage $EC(P)$ quality measure counts the coverage that is exclusive for each pattern, and is defined as follows:

$$EC(P) = \sum_i \left(s(p_i) - \left| S(p_i) \cap \bigcup_{j \neq i} S(p_j) \right| \right)$$

Note that this measure counts the coverage of subgroups that correspond to the codes that contain only a single 1.

DTM Accuracy. Unlike the first two measures, the third quality measure is supervised: it determines the quality of a pattern team on the basis of how well a simple classifier is able to predict the label of individuals, given the patterns as feature set. Finding the optimal pattern team hence amounts to selecting a pattern set by means of a *wrapper approach* [2].

The classifier of choice is the *Decision Table Majority* classifier [6, 8], also known as a *simple decision table*. The idea behind this classifier is to build from the pattern set a contingency table for each possible code, and compute the relative frequency of positive cases for each contingency. For contingencies that do not appear in the database, the relative frequency of positive cases is based on that of the whole database (i.e. the prior). An individual is now classified by computing its code, and returning the majority class within the associated subgroup. This simple approach works surprisingly well, under two conditions: the features (i.e. patterns) have a low cardinality, and the decision table should be based on a relatively small number of features selected from a larger set by means of a wrapper [6]. These conditions clearly hold for our application. The following definition captures the workings of a DTM classifier.

The function f computes a conditional probability estimate for $l(i) = 1$ for a code c, given a set of patterns P:

$$f(P,c) = \frac{|\, S(P,c) \cap S(l(i)=1) \,|}{s(P,c)}, \text{ if } s(P, c) > 0,$$

$$f(P,c) = \frac{s(l(i)=1)}{N}, \text{ if } s(P, c) = 0.$$

The DTM Accuracy $Acc(P)$ quality measure uses the DTM classifier to determine how predictive a pattern set is by computing the accuracy of the classifier by means of cross-validation. This will reduce the risk of choosing a pattern set that over-fits. As a more efficient alternative, one might consider the purity (the accuracy based on in-sample testing) of the DTM classifier as a quality measure. Informal experimentation has shown that results thus obtained are very close to the cross-validated accuracy.

It is important to note that instead of a DTM classifier, any classifier can be applied. Furthermore, obtaining a good classifier is not our primary goal: we are merely using a classifier in order to obtain a well-performing pattern team.

Area Under Curve. The Area Under Curve $AUC(P)$ quality measure computes the area of the convex hull of the patterns in P in ROC-space [1]. The quality measure is computed by plotting the patterns in P in ROC-space, along with the points (0, 0), (1, 0), (1, 1), and computing the area of the convex hull of these $k + 3$ points.

4 Intuitions and Quality Measures

The different quality measures introduced in the previous section capture different aspects of pattern sets. In this section, we examine how these measures fit the intuitions introduced in Section 0. Furthermore, we analyse the correlation between measures, and hence to what extent measures capture similar qualities of pattern sets. As the quality measures are chosen such that they capture at least one intuition perfectly, we can use the correlations between measures to understand how they map to intuitions. If a certain measure is uncorrelated with another measure that is designed to fit a particular intuition, then this first measure cannot be useful for said intuition. The following experiment will support our discussion of quality measures.

Experiment. The database under consideration is the multi-relational database Mutagenesis [5]. It contains structured descriptions of 188 molecules that fall in two classes: mutagenic (66.5%) and non-mutagenic. Although multiple versions of the database exist, with various amounts of information about the molecular structure and properties of the molecules and atoms, we will use a version that contains the basic molecular structure, as well as two numeric attributes on the molecule level (Lumo and LogP). Additionally we have added two aggregated attributes on the molecule level, describing the number of atoms and the number of distinct elements. Hence there is a certain level of redundancy in the database, which may lead to different patterns capturing more or less similar properties of the molecules. Furthermore, the availability of multiple numeric attributes allows for a large range of decision

boundaries, which should lead to redundancy in the patterns, as well as moderate variations of patterns.

Predictive patterns were discovered using Safarii in supervised mode. Subgroups were discovered using the absolute value of the novelty (a.k.a. weighted relative accuracy) interestingness measure, and a minimum support threshold of 5%. Relatively moderate search conditions were used in order to arrive at a manageable result set. The outcome is a collection of 51 multi-relational patterns describing subgroups that show a substantial deviation in mutagenicity, either positive or negative. The original database, as well as the propositionalised version of the 51 binary features can be obtained from the authors.

These results lead to the following conclusions. The results are also summarised in Fig. 1. Each cell describes how useful a particular measure is for a given intuition. If a particular quality measure was defined with a specific intuition in mind, the corresponding cell is coloured grey. For supporting data, see [5].

Joint Entropy. The quality measure *H* clearly captures Intuitions I1 to I3 [4]. If two patterns cover almost the same subgroup, having both of them in the pattern team will not give a significant improvement in uniformity of the distribution over just having one of the two (I1), and hence *H* will favour pattern sets with more diversity among the patterns. The same holds for patterns that follow directly from multiple other patterns (I3). As *H* is insensitive to replacement with complementary patterns, I2 applies. *H* and EC turn out to be uncorrelated, and hence a pattern team optimised with respect to *H* can not be expected to satisfy I4.

Surprisingly, patterns teams optimised for *H* perform reasonably well for classification, despite the unsupervised nature of this measure. Therefore, we can say that *H* captures I5 to a reasonable degree. *H* and *AUC* are uncorrelated, and hence *H* is not a good measure for finding pattern sets on the convex hull in ROC space (I6).

Exclusive Coverage. The measure *EC* penalises overlap, and thus having two similar patterns is unlikely. *EC* therefore satisfies to some extent I1, with the exception of patterns with low support (and thus low penalty) that sometimes appear in copies. I2 and I3 however are not satisfied, because complements and logical redundancy are promoted. Clearly I4 is satisfied. The performance of a DTM classifier is unrelated to the patterns being mutually exclusive: *EC* does not satisfy I5. The same holds for I6.

DTM Accuracy. The measure *Acc* correlates quite well with *H*, and hence captures I1 to I3 moderately well. This should be no surprise, as redundancy in the pattern set cannot benefit the classification score. It turns out that *Acc* in general does not provide mutually exclusive pattern sets (I4). Clearly I5 is satisfied. The experiment shows a very slight correlation between the classification score and the area under curve (I6). At least, poorly performing pattern sets consist of patterns below the convex hull in ROC space. A higher correlation might be expected if only positive patterns would have been produced in the initial discovery phase, as many predictive patterns now appear below the diagonal.

Area Under Curve. As follows from the discussion above, the *AUC* measure is really only useful for I6. A pattern team consisting of patterns on the convex hull apparently is not very useful as input to a classifier. The only purpose of such a pattern

team would therefore be to provide patterns that are optimal individually, rather than as a team.

	Joint Entropy	Exclusive Coverage	DTM Accuracy	AUC
Intuition 1	very high	moderate	high	
Intuition 2	very high		high	
Intuition 3	very high		high	
Intuition 4		very high		
Intuition 5	high		very high	low
Intuition 6			low	very high

Fig. 1. Informal analysis of how well the different quality measures fit the six intuitions

5 Related Work

The domain of feature selection [2, 6, 8] provides good inspiration for pattern filtering techniques, since every pattern in our view can be interpreted as a virtual binary feature. When selecting a feature selection technique, one has to make sure that one or more of our intuitions is satisfied. Many feature selection methods consider the quality of individual features, for example based on correlation with the target concept, and thus potentially produce redundant feature sets. Selection techniques that do consider the value of features in the context of others are more precisely referred to as *feature subset selection* techniques. Wrapper methods are good examples of such techniques [6, 8]. For an overview, see [2].

A domain concerned with the production of a concise set of interesting patterns is known as Subgroup Discovery [1, 3, 9, 10, 12]. The typical approach is to define an interestingness measure, often related to correlation with the target, and then find the top k patterns with respect to this measure. Unfortunately, such techniques often do not consider potential redundancy, and therefore suffer from the same limitations as many feature selection methods. Zimmermann et al. [12] describe a method called CorClass for finding the top k predictive association rules, based on interestingness measures such as novelty, information gain or X^2. The convexity of such measures can be used to find the best rules efficiently. However, due to the redundancy among these rules, relatively high values of k ([12] proposes $k = 1000$) are needed to at least include the essential dependencies required for obtaining good predictive scores. A range of well-known rule combination strategies is used.

For more related work, see [5].

6 Conclusions and Further Work

We have presented a method for reducing the number of patterns returned to the user by a pattern discovery algorithm. The method works by selecting from the (potentially large) collection of patterns deemed interesting by the discovery algorithm a small set of patterns that optimises some quality function for pattern sets. We refer to such an optimal set of patterns of specific size as a *pattern team*. By only allowing a small number of patterns in the pattern set, and selecting the right quality measure, the

resulting pattern team reduces the amount of redundancy between patterns, while retaining as much of the information captured by the patterns as possible. We have presented four measures that capture different qualities of pattern sets. Two unsupervised measures, *Joint Entropy* and *Exclusive Coverage*, promote independence or reduce overlap, respectively. The remaining two supervised measures, *DTM Accuracy* and *Area Under Curve*, are based and how well the pattern set performs as input to a simple classifier, and how well it performs as a collection of points in ROC-space, respectively. Initial experimentation shows that Joint Entropy and DTM Accuracy produce the most useful results, and satisfy a number of intuitive expectations of end-users concerning non-redundancy and predictive quality of the patterns returned.

We have implemented the proposed pattern team discovery scheme in the Safarii system. Space limitations unfortunately prevent a detailed description of algorithmic aspects of the efficient computation of pattern teams. Interesting optimisations over naïve implementations can however be obtained for the quality measures presented [4], and we intend to extend our work in this direction. Furthermore, alternative quality measures can be thought of. Apart from quality measures on the level of pattern sets, one could envisage selecting pattern sets on the basis of inductive queries based on relationships between its member patterns, for example by requiring a certain amount of dissimilarity between every pair of patterns. A pattern team would thus optimise a given quality measure, as well as satisfy a number of inductive constraints.

References

1. Fürnkranz, J., Flach, P., *ROC 'n' Rule Learning – Towards a Better Understanding of Covering Algorithms*, Machine Learning, 58, 39–77, Springer, 2005
2. Guyon, I., Elisseeff, A., *An Introduction to Variable and Feature Selection*, Journal of Machine Learning Research 3, 1157-1182, 2003
3. Knobbe, A.J., *Multi-Relational Data Mining*, Ph.D. dissertation, 2004, http://www.kiminkii.com/thesis.pdf
4. Knobbe, A.J., Ho, E.K.Y, *Maximally Informative* k-*Itemsets and their Efficient Discovery*, in Proceedings of KDD 2006, 2006
5. Knobbe, A.J., Ho, E.K.Y, *Pattern Teams*, long version, 2006, http://www.kiminkii.com/publications.html
6. Kohavi, R., *The Power of Decision Tables*, In Proceedings of ECML '95, 1995
7. Mielikäinen, T., Mannila, H., *The Pattern Ordering Problem*, In Proceedings PKDD 2003, LNAI 2838, 2003
8. Pfahringer, B., *Compression-Based Feature Subset Selection*, In Proceedings of IJCAI '95, 1995
9. *Safarii Multi-Relational Data Mining Environment*, http://www.kiminkii.com/safarii.html, 2006
10. Scheffer, T., Wrobel, S., *Finding the Most Interesting Patterns in a Database Quickly by Using Sequential Sampling*, Machine Learning Research 3, 2002
11. Yan, X., Cheng, H., Han, J, Xin, D., *Summarizing Itemset Patterns: A Profile-Based Approach*, In Proceedings KDD'05, 2005
12. Zimmermann, A., De Raedt, L., CorClass: Correlated Association Rule Mining for Classification, In Proceedings DS 2004, 2004

Compression Picks Item Sets That Matter

Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes

Department of Computer Science
Universiteit Utrecht
{mleeuwen, jillesv, arno}@cs.uu.nl

Abstract. Finding a comprehensive set of patterns that truly captures the characteristics of a database is a complicated matter. Frequent item set mining attempts this, but low support levels often result in exorbitant amounts of item sets. Recently we showed that by using MDL we are able to select a small number of item sets that compress the data well [11]. Here we show that this small set is a good approximation of the underlying data distribution. Using the small set in a MDL-based classifier leads to performance on par with well-known rule-induction and association-rule based methods. Advantages are that no parameters need to be set manually and only very few item sets are used. The classification scores indicate that selecting item sets through compression is an elegant way of mining interesting patterns that can subsequently find use in many applications.

Keywords: frequent item sets, MDL, classification.

1 Introduction

Ever since the first paper on association rule mining [1], mining for frequent item sets has been a popular topic, as it has many useful applications. By now there are many algorithms that discover the frequent item sets efficiently [1,5].

Another problem, however, is far from solved: the explosion of the number of results. Over the years, many solutions have been proposed, e.g., closed [15] and maximal [2] item sets. Most, if not all of these methods can be understood as a compression of the result set, some methods are lossless (closed item sets) while others are lossy (maximal item sets).

Recently, we proposed a radically different solution to this problem [11]. A set of item sets is interesting iff *it yields a good (lossless) compression of the database* rather than a good compression of the set of all frequent item sets. To determine whether or not a subset of the set of frequent item sets yields a good compression of the database, we used the *Minimum Description Length Principle* (MDL) [6].

As shown in [11], heuristic algorithms yield sets of frequent item sets that are easily four orders of magnitude smaller than the complete set of frequent item sets and give high compression ratios. Clearly, the MDL principle indicates that these small sets characterise the database. But, how characteristic are they? That is, do these small sets differentiate between different databases? In this paper we investigate this problem using *classification*.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 585 – 592, 2006.
© Springer-Verlag Berlin Heidelberg 2006

The small set of item sets (well, actually the code table they come from) compresses the original database well. Of course, this compression scheme can be used for all possible transactions. If the code compresses such an arbitrary transaction well, it "belongs" to the database.

This observation suggests a classification algorithm. Split the training database according to class and remove the items indicating the class from all transactions. Then, compute a code table for each of these databases. The set of code tables so derived form a classifier: a transaction t gets assigned to the class C whose code table compresses t best.

The accuracy of this classifier is an independent characterisation of the quality of the small set of item sets our compression based mechanism picks from the huge set available. The experiments of this paper show that this classifier performs on par with well-known rule-induction and association-rule based methods. In other words, compression picks those sets that capture the data characteristics: the patterns that matter.

2 Compression

In this section we give a brief description of our compression based technique to filter out a small set of descriptive item sets from a vast amount of (frequent) item sets; a full description can be found in [11]. To make referring to our compression method easier, we will name it Krimp from now on (which is Dutch for "to shrink").

The first essential element of Krimp is a *code table*. Such a code table has item sets on the left-hand side and codes for these item sets on its right-hand side. The item sets in the code table are ordered descending on 1) item set length and 2) support. The actual codes on the right-hand side are of no importance, but their lengths are.

A transaction t is encoded by Krimp by searching for the first item set c in the left-hand side of the code table for which $c \subseteq t$. The code for c becomes part of the encoding of t. If $t \setminus c \neq \emptyset$, the algorithm continues to encode $t \setminus c$. Since we insist that each coding table contains at least all singleton item sets, this algorithm gives a unique encoding to each (possible) transaction. The set of item sets used to encode a transaction is called its *cover*. Note that the coding algorithm implies that a cover consists of non-overlapping item sets.

The length of the code of an item in a code table CT depends on the database we want to compress; the more often a code is used, the shorter it should be. To compute this code length, we code each transaction in the database db. The *frequency* of an item set $c \in CT$ is the number of transactions $t \in db$ which have c in their cover.

The relative frequency of $c \in CT$ is the probability that c is used to encode an arbitrary $t \in db$. For optimal compression of db, the higher $P(c)$, the shorter its code should be. In fact, we have the optimal code length [6] for c as $-\log(P(c))$.

The length of the encoding of a transaction is now simply the sum of the code lengths of the item sets in its cover. The size of the encoded database is the sum of the sizes of the encoded transactions. For the code table size, we only count those item sets that have a non-zero frequency. The size of the right-hand side column is obvious; it is simply the sum of all the different code lengths. For the left-hand side column, note that the simplest code table possibly consists of only the singleton item

sets. This is the *standard encoding* which we use to compute the size of the item sets in the left-hand side column.

In [11] we defined the optimal set of item sets as that one whose associated code table minimizes the sum of the code table and encoded database sizes. The compression algorithm starts with a valid code table (generally only the collection of singletons) and a sorted list of candidates. These candidates are sorted descending on 1) support and 2) set length. Each candidate is considered by inserting it at the right position in CT and calculating the new compressed size. A candidate is kept in the code table iff the resulting total size is smaller than it was before (Naïve-Compression).

The default Krimp algorithm, Compress-and-Prune, considers each existing code table element for pruning when a new candidate has been added: when an existing element does not contribute to compression it is permanently pruned. For more details, please see [11].

3 Classification

As usual in data mining, we assume that our database of transactions is an i.i.d. sample from some underlying data distribution. The result of any data mining algorithm is only useful if it reflects structure in the underlying distribution rather than spurious structure in the sample. Translated to Krimp, this means that we expect Krimp to compress an arbitrary transaction sampled from the underlying distribution well.

To make this more precise, assume that our code table CT has no zero-frequency item sets. Let t be an arbitrary transaction over the items $\mathcal{I}$, then:

$$\begin{aligned} l_{CT}(t) &= \sum_{c \in \text{cover}(t)} l_{CT}(c) = \sum_{c \in \text{cover}(t)} -\log(\text{P}(c \mid db)) \\ &= -\log\left(\prod_{c \in \text{cover}(t)} \text{P}(c \mid db)\right) = -\log(\text{P}(t \mid db)) \end{aligned} \tag{1}$$

The last equation is a Naïve Bayes [4] like assumption: we assume that the item sets that cover a transaction are independent. Clearly, this is overly optimistic as item sets in a cover are not allowed to overlap! However, Naïve Bayes is known to perform well, even if the independence assumption is violated. Therefore, we ignore this violation for the moment.

Now, assume that we have two databases db_1 and db_2 generated from two different underlying distributions. We apply Krimp to both databases and get two code tables, CT_1 and CT_2. We are given a new transaction t that is generated under either the distribution for db_1 or the one for db_2, but we are not told which one. How do we decide to which distribution t belongs? Under the Naïve Bayes assumption, we have:

$$l_{CT_1}(t) < l_{CT_2}(t) \rightarrow \text{P}(t \mid db_1) > \text{P}(t \mid db_2) \tag{2}$$

Hence, the Bayes optimal choice is to assign t to the distribution that leads to the shortest code length. We already know that the result of Krimp is a small set of item sets that is optimal with regard to the MDL principle. The above observation suggests an independent way to assess the quality of this result: how well does it classify?

The construction of the Krimp classifier works as follows:

1. Split the training database according to class
2. Remove the items that indicate the class from each transaction
3. Compress each of the databases, yielding a code table CT_i for each class C_i

Then, to classify an unseen transaction t:

1. Compute $l_{CTi}(t)$ for all classes C_i
2. Assign t to the class that minimizes $l_{CTi}(t)$

Note that we do a Laplace correction during classification. That is, all usage frequencies are increased by one, ensuring valid code lengths for zero frequency sets. We generate code tables at fixed support intervals *report-sup* during compression, and use these for classification. To pair the code tables for classification we use two methods: absolute and relative. In absolute pairing, the code tables that have been generated at the same support levels are matched. Relative pairing matches code tables of the same relative support between 100% and 1% of *max-sup* (per class).

4 Advanced Classifiers

Many algorithms that can be used for classification have been proposed, many of which fall into either the class of rule-induction-based or that of association-rule-based methods. Because we use classification as a quality measure for the patterns that Krimp picks, we will compare our results with those obtained by some of the best existing classifiers. Comparison can be done with rule-induction-based methods such as C4.5 [9], FOIL [10] and CPAR [14], but we are more interested in an in-depth comparison with association-rule-based algorithms like iCAEP [16], HARMONY [12], CBA [7] and LB [8]. We believe this comparison is more interesting because these methods also use a collection of item sets for classification. Because we argued that our method is strongly linked to the principle of Naïve Bayes (NB) [4] its imperative we also compare to this method. Because these methods were devised with the goal of classification in mind, we would expect them to outperform the Krimp classifier. The goal of Krimp is extracting a small set of interesting patterns.

5 Experiments

In order to assess the quality of Krimp's data distribution approximation, we tested on a plethora of UCI databases and compared accuracies to those obtained by a large range of existing classification algorithms. As the algorithm currently only deals with item sets, we used discretised versions [3,5] of the databases.

All experimental results were obtained using 10-fold cross-validation. We report the *min-sup* thresholds we used for each dataset. We compare to scores taken from the publications in which the respective classifiers were described [8,12,16]. These all used the same discretised datasets. The missing scores for Naïve Bayes and C4.5 have been acquired using Weka [13].

Although virtually any collection of item sets could be considered as candidates, our focus has been on using all frequent and closed frequent item sets. Figure 1 shows that using either all or closed frequent item sets and Naïve-Compression or Compress-and-Prune hardly influences classification. Because the reduction of the number of item sets is largest with all frequent item sets and on-the-fly pruning [11], we use this combination in all experiments presented in the rest of this section.

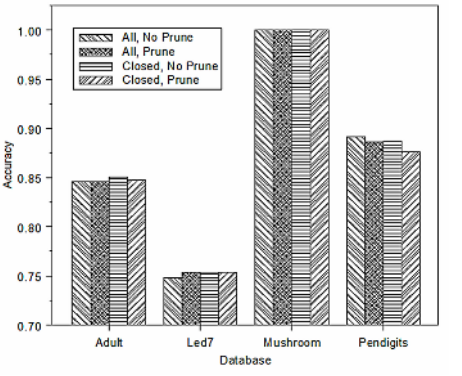


Fig. 1. Accuracies for different candidate sets (all/closed frequent item sets) and pruning enabled/disabled, for 4 datasets

Figure 2 shows that better compression generally leads to better classification. This is not always the case though: for the mushroom dataset, a drop in total compressed size always results in a change in accuracy, but this change is not always positive. The drops in accuracy could have several causes, the most likely being that item sets characteristic for both classes are added to the code tables at that support level. This would make the encoded probability distributions look more alike and make discrimination more difficult. The longer candidates with lower support make an important difference between the classes though, as a 100% classification score is obtained later on.

In case of the mushroom dataset, code tables for the two classes are paired absolute, e.g. having the same minimum support. This is not possible for the anneal dataset, as it consists of 5 classes that have a very skewed a priori distribution. However, using relative code table pairing, competitive scores can be obtained for anneal. The

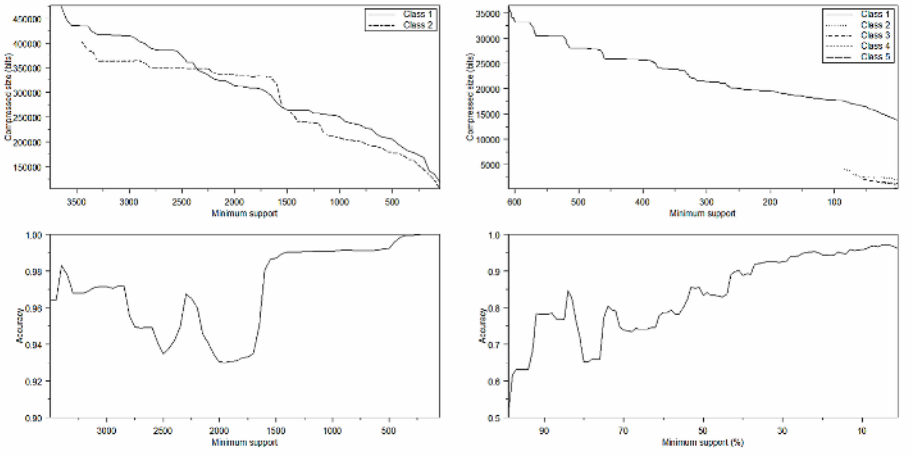


Fig. 2. Minimum support vs total compressed size per class and accuracy for mushroom (absolute pairing, left) and anneal (relative pairing, right)

Table 1. Statistics on 8 UCI datasets using Compress-and-Prune with all frequent item sets as candidates. Numbers of candidates, resulting code table sizes |CT| (excluding singleton sets) and compression ratios at given min-sups are summed for all classes and 10-fold cross validated. The best 10-fold cross validated accuracies are given (not always belonging to *min-sup*).

Dataset	#rows	|C|	|$\mathcal{I}$|	#candidates	|CT|	Compr ratio	Acc %	min sup
Adult	48842	2	97	1679483	994	3.40	84.6	50
Anneal	898	6	71	2117941	110	2.47	97.0	1
Ionosphere	351	2	157	42908227	107	1.47	90.6	35
Led7	3200	10	24	12815	720	3.49	75.3	1
Mushroom	8124	2	119	92163948	391	3.94	100.0	50
Pendigits	10992	10	86	255838056	1996	2.19	88.6	1
Waveform	5000	3	101	942271	741	1.71	77.2	100
Wine	178	3	68	1049213	167	1.26	97.7	1

accuracy graph emphasizes that better compression doesn't always result in better classification: the highest accuracy is achieved shortly before *min-sup* is reached. This might be caused by slight overfitting.

In Table 1 we provide an overview of the achieved compression together with classification scores of Krimp on a variety of UCI datasets. The numbers clearly indicate that Krimp copes with a wide range of datasets and always strongly reduces the number of candidates. The reduction of the number of item sets is enormous, up to 5 orders of magnitude (in case of PenDigits, the beastly amount of 255 million item sets is cut back to only 1996!). Looking at the *min-sups*, only waveform seems like a strange outlier: because of its characteristics and density, compressing is computationally very intensive, despite the seemingly small numbers of rows and candidates.

As compression proceeds and classification improves, average encoded transaction lengths also change. In general, the difference between the length of the correctly (incorrectly) winning class and that of the losing class increases (decreases). This means that this difference could be used as uncertainty factor: a larger difference results in a higher certainty of assigning the correct class. (Not shown due to space limitations.)

The comparison of classification accuracies in Table 2 shows that the Krimp code tables are of such high quality they can compete with many methods specifically built for classification. Our elegant though make-shift MDL based classification extension

Table 2. Accuracy scores on 10 UCI datasets, 10-fold cross validated. Scores taken from [16], missing scores for Naive Bayes and C4.5 obtained using Weka [13].

Dataset	#cl	#rows	iCAEP	NB	LB	CBA	C4.5	Krimp	min sup
Adult	2	48842	80.9	82.7	85.1	75.2	**85.5**	84.6	50
Anneal	6	898	95.1	86.3		**98.1**	90.4	97.0	1
Breast (Wisc)	2	699	**97.4**	96.0	96.9	95.3	85.4	94.1	1
Iono	2	351	90.6	82.6		**92.1**	91.5	90.6	35
Iris	3	150	93.3	**96.0**		92.9	84.7	**96.0**	1
Mushroom	2	8124	99.8	95.8			**100.0**	**100.0**	20
Pima	2	768	72.3	74.7	**75.8**	73.1	72.5	75.0	1
Tic-tac-toe	2	958	92.6	69.6		**100**	84.6	87.1	1
Waveform	3	5000	81.7	80.0	79.4	75.3	75.1	77.2	100
Wine	3	178	**98.9**	96.6		91.6	93.8	97.7	1

Table 3. Accuracy scores on 8 large UCI databases, 10-fold cross validated. Scores taken from [12]. *Min-sup* for Harmony set to 50.

Dataset	#cl	#rows	FOIL	CPAR	SVM	Harmony	Krimp	min sup
Adult	2	48842	82.5	76.7	84.2	81.9	84.6	50
Led7	10	3200	62.3	71.2	73.8	74.6	**75.3**	1
LetterRecog	26	20000	57.5	59.9	67.8	**76.8**	68.1	50
Mushroom	2	8124	99.5	98.8	99.7	99.9	**100.0**	20
Nursery	5	12960	91.3	78.5	91.4	**92.8**	92.4	1
PageBlocks	5	5473	91.6	76.2	91.2	91.6	**96.6**	1
PenDigits	10	10992	88.0	83.0	93.2	**96.2**	88.6	1
Waveform	3	5000	75.6	75.4	**83.2**	80.5	77.2	100

is always at the head of the pack and delivers two wins. Note that on average Krimp performs better than Naïve-Bayes, although it is based on the same assumption. Unfortunately no further results for LB and CBA were available.

When comparing on some of the largest UCI datasets in Table 3, Krimp proves to do exceptionally well. The bigger datasets allow MDL to better work its magic, making a good selection of which sets to include in its code tables. Overall, Krimp performs very well, as it is always close to the best scores and puts down the best score a number of times.

6 Discussion

Outstanding classification was not the ultimate goal of the experiments presented in the previous section, as explained before. We are very content as our explicit intention was to verify that our method is very capable at representing data distributions with only few item sets. The results of the experiments clearly verify this hypothesis.

Although Krimp has not been designed for classification and no attempts have been made to enhance the differences between code tables for different classes, the results show that it performs well compared to the best known classifiers. As the mentioned association-rule-based classifiers also select item sets that characterise the classes, it is interesting to compare our method with those in a qualitative way.

The selection method iCAEP uses is far less effective than MDL: the amount of Emergent Patterns may grow enormously and a lot of item sets may be required in the end. Also, Krimp is independent of the base distribution, iCAEP is not. Large Bayes does succeed in selecting only a small set of item sets that is used to determine a class-based probability distribution, but it uses an interestingness measure that requires a parameter that needs to be chosen manually.

HARMONY selects at most one rule per transaction in the training database. Although it is likely that equal rules are selected and therefore merged, the final set of rules can still be quite large. In Krimp, an item set is only used if it helps to compress the whole training database; we therefore believe that HARMONY is more prone to noise and overfitting than our method and the rules do not represent the data as well.

The elegancy of the classifier based on Krimp lies in 1) the use of only MDL for building and applying the classifier, 2) the small amount of item sets required and 3) the natural way it deals with multi-class problems and skewed class distributions.

7 Conclusion

Krimp picks the item sets that matter. From staggering amounts it selects only handfuls of item sets that not only attain high compression ratios, but can compete with today's cutting edge classifiers as well. We therefore conclude that Krimp is well suited for capturing the characteristics of the data.

As an independent measure of the quality of the selected item sets we used classification. The training data is compressed per class, the resulting code tables are used to encode new test instances. Following Bayes optimal choice, the class with the code table assigning the shortest code has the highest probability of being the correct data distribution and is therefore chosen winner. Classification accuracies achieved are on par with the best known classifiers.

Not only is the selected collection of item sets of high quality, the reduction of the number of candidates is huge, generally many orders of magnitude. Krimp thus proves to be a generic method that finds small sets of patterns that encapsulate the probability distribution of the data well.

References

1. Agrawal, R., Imielinksi, T. & Swami, A. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD conference*, 207-216 (1993).
2. Bayardo, R. Efficiently mining long patterns from databases. In *Proc. ACM SIGMOD conference*, 85-93 (1998).
3. Coenen, F. The LUCS-KDD Discretised/normalized ARM and CARM Data Library. In *www.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets/dataSets.html*.
4. Duda, R. & Hart, P. *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York (1973).
5. Goethals, B. et al. FIMI website. In *fimi.cs.helsinki.fi*
6. Grünwald, P.D. Minimum description length tutorial. In Grünwald, P.D., Myung, I.J. & Pitt, M.A., editors, *Advances in Minimum Description Length*. MIT Press (2005).
7. Liu, B., Hsu, W. & Ma, Y. Integrating Classification and Association Rule Mining. In *Proc. KDD-98*, New York (1998).
8. Meretakis, D. & Wüthrich, B. Extending Naïve Bayes Classifiers Using Long Itemsets. In *Proc. of the Conference on Knowledge Discovery and Data Mining* (1999).
9. Quinlan, J.R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann (1993).
10. Quinlan, J.R. FOIL: A Midterm Report. In *Proc. ECML '93* (1993).
11. Siebes, A.P.J.M., Vreeken, J. & Van Leeuwen, M. Item Sets That Compress. In *Proc. of the ACM SIAM Conference on Data Mining* 393-404 (2006).
12. Wang, J. & Karypis, G. HARMONY: Efficiently Mining the Best Rules for Classification. In *Proc. of the ACM SIAM Conference on Data Mining* (2005).
13. Witten, I.H. & Frank, E. *Data Mining: Practical machine learning tools and techniques*, 2nd edition, Morgan Kaufmann, San Francisco (2005).
14. Yin, X. & Han, J. CPAR: Classification based on Predictive Association Rules. In *Proc. SDM '03* (2003).
15. Zaki, M.J. & Orihara, M. Theoretical foundations of association rules. In *Proc. ACM SIGMOD workshop on research issues in KDD* (1998).
16. Zhang, X., Guozhu, D. & Ramamohanarao, K. Information-based Classification by Aggregating Emerging Patterns. In *Proc. IDEAL* (2000).

Discovering Overlapping Communities of Named Entities

Xin Li[1], Bing Liu[1], and Philip S. Yu[2]

[1] Department of Computer Science, University of Illinois at Chicago, 851 S. Morgan Street, Chicago, IL 60607-7053
{xli3, liub}@cs.uic.edu
[2] IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532
psyu@us.ibm.com

Abstract. Although community discovery based on social network analysis has been studied extensively in the Web hyperlink environment, limited research has been done in the case of named entities in text documents. The co-occurrence of entities in documents usually implies some connections among them. Investigating such connections can reveal important patterns. In this paper, we mine communities among named entities in Web documents and text corpus. Most existing works on community discovery generate a partition of the entity network, assuming each entity belongs to one community. However, in the scenario of named entities, an entity may participate in several communities. For example, a person is in the communities of his/her family, colleagues, and friends. In this paper, we propose a novel technique to mine overlapping communities of named entities. This technique is based on triangle formation, expansion, and clustering with content similarity. Our experimental results show that the proposed technique is highly effective.

Keywords: Community of named entities, community mining.

1 Introduction

Knowledge discovery in social networks has attracted a great deal of attention due to its successful application in Web search engines. PageRank [2] and HITS [9] are two representative Web page ranking algorithms. Both algorithms regard each Web page as an entity in the social network, and each hyperlink is a relationship between the entities. In addition, HITS discovered that there exist multiple Web communities among relevant pages when the query term has several meanings.

Going beyond the hyperlinked Web environment, we believe that communities also exist among named entities in text documents. In the Web, there are explicit links connecting entities. However, such links do not exist in free text documents. In this work, we consider that named entities are implicitly linked if they co-occur in the same sentence.

Our objective in this work is to discover overlapping communities of named entities, i.e. the names of persons, organizations, from Web contents and text documents. Our research is motivated by two major factors.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 593 – 600, 2006.
© Springer-Verlag Berlin Heidelberg 2006

1. Named entity terms are among the most frequently searched terms on the Web. Based on a report from Yahoo! in 2005[1], all the top ten search terms are named entities. For those frequently searched entities, users' interests can be diverse. By finding the overlapping communities, we can separate the various facets about the entity of interest.
2. Named entities are natural actors according to the definition of social networks [13]. The original concept of social network was proposed to study social relationships among people and organizations. By automating data analysis from vast volume of texts, we can analyze social network at a grand scale.

Although many community-mining algorithms exist, we are unable to use them for our purpose because they are mainly partitioning algorithms [7][12] that do not allow the same entity to appear in multiple communities. In contrast, an entity belongs to multiple communities in most of realistic social networks.

Given a named entity, our algorithm works as follows. It first collects a set of relevant documents on the named entity. All the entities co-occurring in the sentences are linked together to generate a named entity graph. We also keep the contextual information, which are noun terms in the co-occurrence sentences. The algorithm then identifies community cores, and clusters those fringe members into the cores.

2 Related Work

The work on community structure discovery on the Web first appeared in the HITS algorithm [9]. Since then the issue of community discovery has been studied in a variety of environments. However, we are not aware of any work on extracting communities of named entities from text documents at the time of paper submission.

[7] proposed a Web community mining algorithm based on Max flow-Min cut. In [12], a partitioning algorithm was also proposed, so does [3] but in the email context. In [5], the authors studied the community issue in a graph from a local perspective. They introduced the concept of "curvature" for each vertex v to measure how well connected v's neighborhood is. The authors made an observation that a community expands mainly by triangles sharing a common edge. The same observation was also made in [12].

In addition to the Web community issue, other works studied the community structure from other aspects. [4] applied the concept of community in the Word Sense Disambiguation problem. Link analysis also has other applications, such as group membership detection [10] and text summarization [6].

Another related research focused on extracting binary relations from the Web. In [1], the author designed an algorithm to find a large number of book/author pairs from only several seeds. [8] extracts relations of named entities from a large text corpus. It groups relations of entities according to their text similarity. The work was not concerned with communities because similar relations do not mean that the entities involved are in the same community.

[1] http://tools.search.yahoo.com/top2005/

3 Problem Definition

This section defines communities, and the objective of this work.

Definition (Community): Given a finite set of entities $S = \{s_1, s_2, \ldots, s_n\}$, a community is a pair $C = (T, G)$, where T is a *theme* and $G \subseteq S$ is a subset of S that shares the theme T. If $s_i \in G$, s_i is called a *member* of the community C. If $C = \emptyset$, C is an *empty community*.

A theme defines a community. Given a theme T, the set of members of the community is uniquely determined. Thus, two communities are equal if they have the same theme. A theme can have a variety of forms: it can be an event or a concept.

An element s_i in S can be in any number of communities, i.e. multiple communities may share members. We denote that an entity associates with a set of themes by s_i:$\{T_1, T_2, \ldots, T_m\}$, where T_k is a theme of community C_k, to which entity s_i belongs.

Given a data set, which can be a set of Web pages, emails, or text documents, usually there is no metadata regarding community available. The system needs to discover the hidden community structure from the linkage among entities. The forms that communities manifest themselves may vary.

Web pages. Web page authors sharing common interests often cite others' pages through hyperlinks. Members in a Web community are more likely to be linked with their peers than pages outside the community. The text from those community member pages can be used to extract the community theme.

Emails. Members of a community are more likely to communicate with one another. The email contents of the community provide a good summary of the community theme.

Text documents. Named entities within a community are more likely to appear together in the same sentence. The words in those co-occurrence sentences reflect community themes.

The key form of community manifestation is that its members are "linked" to each other in some sense. Such links indicate that they share a common theme. Given a data set containing named entities, our objective in this work is to discover the hidden communities of the named entities, and identify the community themes.

4 Mining Overlapping Named Entity Communities

There are two main tasks in discovering named entity communities from documents. The first one is to acquire named entity relationships; the second task groups named entities into different communities based on their relationships and the text contents.

4.1 Finding Entity Relationships

Given a named entity, the system first searches the Web, blogs or a document collection to find those relevant documents. It then uses a named entity parser MINIPAR [11] to tag the named entities in sentences. Furthermore, each sentence that contains at least two named entities of same type is extracted. All entities in a sentence are considered to be connected pair-wise with an edge.

After all documents are processed, a set of distinctive edges is produced. We attach a *strength* to each edge, which is computed using mutual information. In our case, the mutual information reflects the closeness of two entities. Let the entities of an edge be a and b, and $\Pr(a, b)$ be the co-occurrence probability of (a, b). If the total co-occurrences of all edges is N, and there are n co-occurrences of a and b, then $\Pr(a, b) = n/N$. Let $f(a)$ and $f(b)$ be the probabilities of occurrences of a and b respectively in the edges. The mutual information is defined as follows:

$$I(a,b) = \Pr(a,b) \log_2 \frac{\Pr(a,b)}{f(a) f(b)} \quad (1)$$

4.2 Mining Communities

Our community-mining algorithm is a core-periphery clustering algorithm. First, we find all cohesive community cores based on the graph topology. After the formation of community cores, we exploit the content information of the relationships within each community core, and group the peripheral entities with the community cores to obtain the final communities.

The basic building blocks of community cores in our algorithm are triangles. A triangle is a complete graph itself, and is a component of larger complete graphs. It was observed in [5][12] that a community expands predominantly by triangles sharing a common edge.

Our community-mining algorithm consists of three major steps.

Finding Triangles. A triangle is formed by edges connecting three entities. It is defined as follows:

Triangle: In a graph $G = (V, E)$, V is a set of vertices, and E is a set of edges among V. For vertices $a, b, c \in V$, if edges $(a, b) \in E$, $(a, c) \in E$, and $(b, c) \in E$, we say vertices a, b, c form a triangle if each edge has at least τ instances and a positive mutual information. τ is a parameter.

Finding Community Cores. An ideal community core is a complete subgraph, i.e., a clique, consisting of a set of vertices such that each pair of vertices is directly connected by an edge. However, this definition is too strong for practical use because the data may be incomplete. We thus relax this definition and give an operational definition.

Community core: Two candidate cores c_1 and c_2 are merged to form a larger community core if there are at least one triangle from c_1 and one triangle from c_2 that share a common edge and form a complete graph of 4 vertices. The resulting community core satisfies the criterion that each vertex in the core is adjacent to three or more other vertices within the same core. We illustrate the definition with the following example. Fig. 1 shows two candidate cores, where c_1 is a 4-vectex core and c_2 is a triangle (which is a smallest core). Since the triangle CBD in c_1 and the triangle BDE in c_2 shares a common edge. If the link CE exists, BCD and BDE form a 4-vectex complete graph BCDE. Therefore, we can join c_1 and c_2 to produce c_3.

The core expansion algorithm works as follows. T is an array of triangles. For each triangle, a core c consisting of only the triangle is created. The algorithm tries to

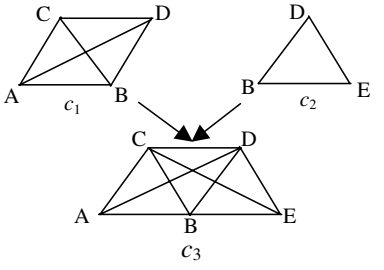


Fig. 1. An example of community core

merge these candidate cores by checking every triangle pair to see whether they can join. If the triangle $T[i]$ can join with $T[j]$, then their cores are merged together to form a larger core.

Clustering around Community Cores. Our next task is to group the remaining triangles and edges with the community cores. We exploit the content information of the community cores. If a triangle or an edge has a high content similarity, which is measured by text similarity, with a community core, it indicates that they are likely to share a common theme. Consequently, it will be clustered into that community.

Let the set of *cores* be $C = \{c_1, c_2, \ldots, c_k\}$, and the remaining elements be $S = \{s_1, s_2, \ldots, s_m\}$, which include both triangles and edges. Those elements could not be merged to the cores form their own communities of smaller sizes.

The algorithm first compares the similarity between each element s_i in S with each core. It then adds s_i to the core that has the highest content similarity with s_i. If s_i has 0 similarity with every core, s_i forms a small community by itself. The similarity function between triangles is described below.

triangleSimilarity(s_i, c_j) computes the similarity of a triangle s_i and a core c_j. This similarity is the largest similarity between the triangle s_i and triangle members in c_j that share an edge with s_i. If a triangle s_i does not have a common edge with any triangle in the core, the similarity is 0.

The similarity between two triangles is computed as follows: If they do not share any edge, then their similarity is 0. If they share an edge, their similarity is computed like this: Let the two triangles be t_1 and t_2. t_1 has three edges $\{e_a, e_b, e_c\}$, and t_2 has three edges $\{e_d, e_f, e_c\}$. e_c is the common edge. To calculate the triangle similarity between t_1 and t_2, we combine all the keywords in the edges e_d and e_f together to form a vector $v_{d,f}$, and combine all the keywords of edges e_a and e_b to form a vector $v_{a,b}$. The cosine similarity, which is the standard similarity measure in information retrieval, between the two term vectors is the triangle similarity. In the same way, we can compute the similarity between an edge and a community core.

5 Empirical Evaluation

This section evaluates the proposed technique. We first describe the test documents used. They come from different sources, as we want to test if the proposed algorithm is generally applicable. Our first document collection is from top 500 Web pages

retrieved through the Google search engine for a given entity. The other two document collections are top 1000 relevant documents from Google blog search, and top 300-500 relevant documents from Financial Times (FT) corpus.

Table 1 shows our experiment results from the Web pages. Column 1 gives the name of each entity. Column 2 shows the community ID. Column 3 lists entities for each community sorted in descending order of their degree centrality scores. Due to the space limit, we used the initials for the first names in Table 1. We automatically extracted the top nouns from community context and listed them in the column 4. We also manually added some remarks on the discovered communities in column 5.

To evaluate community members, we manually checked the co-occurrence sentences extracted from original text documents. If an entity member in the discovered community is related to the community theme, we mark the entity member as correct. In Table 1, we used italic font for incorrect entity members. Among n members extracted for community c, there are m correct members; the community accuracy of c is $A(c) = m/n$.

Let us now look at the communities of "Bill Clinton". We can see that both communities B1 and B2 contain very relevant persons. While B3 is much smaller, it also contains the family topic. In the Cheney's communities, we would like to point out that "Mary Cheney" and "Lynne Cheney" are grouped into both political and family communities. In fact, both entities are legitimate members, and play different roles in the two communities. These highlight the key feature of our algorithm, mining overlapping communities.

Table 1. The Discovered Named Entity Communities from Web Pages

Entities	ID	Community members	Summary Terms	Remarks
Bill Clinton	B1	Bill Clinton, G. Bush, H. Clinton, J. Kerry, K. Starr, J. Edwards, A. Gore, J. F. Kennedy, R. Reagan, B. Dole, F. Roosevelt, R. Nixon, N. Gingrich, *D. Rather*, D. Cheney, J. Carter, *J. Lehrer*, V. Foster, R. Perot, S. Hussein, B. Laden, D. Morris, M. Beschloss, W. J. Clinton, *T. Jefferson*, *M. Moore*.	president, election, stage, state, senator, campaign	Political community
	B2	Bill Clinton, P. Jones, M. Lewinsky, K. Starr, L. Tripp, J. Reno, K. Starr, W. J. Clinton, G. Flowers, R. Wright, K. Willey, D. Kendall, *L. Johnson.*	case, president, testimony, lawsuit, jury, deposition	The scandal
	B3	Bill Clinton, Hillary Clinton, Chelsea Clinton.	daughter, wife, time,	Family
Dick Cheney	D1	Dick Cheney, G. W. Bush, J. Kerry, S. Hussein, J. Edwards, C. Powell, B. Clinton, L. Cheney, B. Laden, R. Reagan, G. Ford, R. Clarke, R. Cheney, *T. Russert*, M. Cheney, M. Daniels, A. Gore, P. Leahy, D. Rumsfeld, R. Nixon, P. Wolfowitz, J. Lieberman, *H. Chavez*, *J. Nichols*, D. Quayle, P. Goss, *J. Marshall*, J. Wilson, B. Scowcroft, N. Schwarzkopf, A. Williams, R. Perle, Bush Sr, E. Olson, F. Olson, R. Armitage, T. Ridge, N. Mandela, J. Miller.	president, Iraq, war, administration, defense, secretary	Political community
	D2	Dick Cheney, L. Cheney, M. Cheney, L. A. Vincent, L. Cheney.	daughter, wife, child, issue, family	Family

In the Table 2, the communities of "Tom Cruise" were extracted from Weblog data. We can observe that T1 and T2 are strong communities. To our surprise, T3 was

Table 2. The Discovered Named Entity Communities from Blogs

Entities	ID	Community members	Summary Terms	Remarks
Tom Cruise	T1	T. Cruise, B. Shields, K. Holmes, M. Lauer, O. Winfrey, L. R. Hubbard, K. Preston, *J. Fox*, B. bush, *N. Yan*, *M. Jackson*, S. Johansson, D. Miscavige, M.Rogers, P.Kingsley, L. A. Devette, *B. Pitt*, J. Travolta.	scientology, depression, actress, love, show, paxil	Scientology & psychiatry
	T2	Tom Cruise, K. Holmes, N. Kidman, M. Rogers, P. Cruz, C. Klein, S. Vergara, R. Thomas.	actress, relationship, girlfriend, love, marriage, thing	Dating life
	T3	S. Spielberg, T. Cruise, J. Maguire	war, director, film, year, movie, world	Movies
	T4	R. Hubbard, J. Rodriguez, K. Holmes	adviser, interview, member, scientology	Katie& Scientology
Angelina Jolie	A1	A. Jolie, B. Pitt, Maddox, Z. M. Jolie, J. Aniston, B. B. Thornton, J. Voight, G. Clooney, L. Dern, *L. Croft, King N. Sihamoni*, J. L. Miller.	child, son, people, divorce, love, marriage	Private Life
	A2	A. Jolie, Good Shepherd, Ro. De Niro, M. Damon	drama, cia, history, thriller, universal	Movie Project

a weak community. It indicates that not many blogs paid attention on his movie release. Similarly, T4 was also relevant, but a weak community. The community of "Angelina Jolie" shows the same pattern. Whereas both communities are valid, the private life community is larger than the movie community.

Table 3. The Discovered Named Entity Communities from the FT Newswire Corpus

NE	ID	Community members	Summary Terms	Remarks
Sony	S1	Sony, CBS Records, CBS, MCA, Matsushita, Columbia Pictures, *GE.*	1988, acquisition, purchase, company, year, chairman	Acquisition events
	S2	Motorola, Sony, Apple Computer.	product, media, general, magic, technology, company	Cooperation events
	S3	Sony, Warner Bros., Time Warner, Paramount.	producer, contract, movie, Time, Warner, company	Media companies
	S4	Sony, Toshiba, Panasonic, JVC, Fujitsu, NEC, *IBM*, Hitachi.	Japan, company, electronics, USA, phone, industry	Japanese companies
IBM	I1	IBM, Toshiba, NEC, Microsoft, Fujitsu, Intel, Hitachi, Groupe Bull, HP, Motorola, Apple Computer, NCR, Dell, Sony, Novell, *GM, Nasdaq*, TI, *Time Warner*	PC, computer, company, chip, market, software	PC makers
	I2	IBM, Sun, Groupe Bull, HP, MIPS.	workstation, market, RISC, competition, deal, technology	Workstation makers

We used a newswire corpus in the last experiment. The results in Table 3 further demonstrate the effectiveness of our algorithm. Taking "Sony" as an example, community S3 lists its peer companies in the entertainment business, and S4 contains its peer Japanese companies. Communities I1 and I2 are also interesting. While there is a considerable overlapping between the workstation and PC makers, the link context

reveals two distinct community themes. The accuracy for community extraction from these six entities is 172/193 = 89.1%.

6 Conclusion

This paper studied the problem of mining named entity communities from text documents. So far little work has been done to investigate this issue. By exploiting the named entity co-occurrence, we mapped text documents into a named entity graph. An effective mining algorithm was proposed to mine overlapping communities using triangle expansion and content similarity. We applied our algorithm on a variety of document collections. Our experimental results show that the algorithm is able to discover interesting communities. This work is potentially useful to enhance the Web search related to named entity queries.

References

1. Brin, S. Extracting patterns and relations from the World Wide Web. In Selected papers from the International Workshop on the World Wide Web and Databases (1999).
2. Brin, S. and Page, L. The anatomy of a large-scale hypertext Web search engine. In Proceedings of the seventh international conference on World Wide Web 7.
3. Diesner J. and Carley K.M. Exploration of Communication Networks from the Enron Email Corpus. In Workshop on Link Analysis, Counter-terrorism and Security at the SIAM Data Mining Conference (Newport Beach, California, 2005).
4. Dorow, B. and Widdows, D. Discovering corpus-specific word senses. In EACL, (Budapest, Hungary, 2003), 79-82.
5. Eckmann, J., and Moses, E. Curvature of co-links uncovers hidden thematic layers in the World Wide Web. In Proceedings of the National Academy of Sciences.
6. Erkan, G. and Radev, D. Lexrank: Graph-based centrality as salience in text summarization. In Journal of Artificial Intelligence Research (2004), 22:457-479.
7. Flake, G. W., Lawrence, S., and Giles, C. L., and Coetzee, F. Self-Organization and Identification of Web Communities. In IEEE Computer (2002), 35(3): 66-71.
8. Hasegawa, T., and Sekine, S., and Grishman, R. Discovering Relations among Named Entities from Large Corpora. In ACL (2004), 415-422.
9. Kleinberg, J. Authoritative sources in a hyperlinked environment. In Proceedings of ACM-SIAM symposium on discrete algorithms, (1998) 668 - 677.
10. Kubica, J., Moore, A., Schneider, J., and Yang, Y. Stochastic Link and Group Detection. In Proceedings of the Eighteenth National Conference on Artificial Intelligence.
11. Lin, D. PRINCIPAR-An Efficient, broad-coverage, principle-based parser. In Proceedings of the 15th conference on Computational linguistics, (Kyoto, Japan, 1994).
12. Toyoda, M. and Kitsuregawa, M. Creating a Web community chart for navigating related communities. In Proceedings Hypertext-2001 (Århus, none, Denmark), 103-112.
13. Wasserman, S. & Faust, K. (1998) Social Network Analysis: Methods and Applications.

Closed Non-derivable Itemsets

Juho Muhonen and Hannu Toivonen*

Helsinki Institute for Information Technology
Basic Research Unit
Department of Computer Science
University of Helsinki
Finland

Abstract. Itemset mining typically results in large amounts of redundant itemsets. Several approaches such as closed itemsets, non-derivable itemsets and generators have been suggested for losslessly reducing the amount of itemsets. We propose a new pruning method based on combining techniques for closed and non-derivable itemsets that allows further reductions of itemsets. This reduction is done without loss of information, that is, the complete collection of frequent itemsets can still be derived from the collection of closed non-derivable itemsets. The number of closed non-derivable itemsets is bound both by the number of closed and the number of non-derivable itemsets, and never exceeds the smaller of these. Our experiments show that the reduction is significant in some datasets.

1 Introduction

Itemset mining often results in a huge amount of itemsets. Unfortunately the result usually contains a large number of redundant itemsets. Redundancy is inherent in the collection of all frequent itemsets and is the result of the fact that many itemsets are uninformative if the user is already aware of some other itemsets in that collection. In technical terms, an itemset can be considered redundant if its support can be inferred from other itemsets. Removing redundant itemsets produces a condensed representation of all frequent itemsets.

The best known condensed representations are closed itemsets [1] (or generators/free sets) and non-derivable itemsets [2]. The collection of non-derivable itemsets often is smaller than the collection of closed sets, but given an arbitrary dataset either one may contain fewer itemsets. In this paper we propose a method that combines the ideas of closed and non-derivable itemsets, and is guaranteed to be at least as efficient as the better of the two, while retaining the cabability of losslessly recovering the full collection of frequent itemsets.

2 Basic Concepts from Related Work

The frequent itemset mining problem can be described as follows [3]. We are given a set $\mathcal{I}$ of items and a dataset (multiset) $\mathcal{D}$ of subsets of $\mathcal{I}$ called transactions. A

* Currently at University of Freiburg; supported by Humboldt foundation.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 601–608, 2006.
© Springer-Verlag Berlin Heidelberg 2006

set X of items is called an itemset. The support $s(X)$ of X is the number of transactions that contain X. An itemset is called frequent if its support is no less than a given minimum support threshold δ. We denote the collection of all frequent itemsets by $Freq$. In the rest of the paper, we work on collections of frequent itemsets, but often drop "frequent" for lingvistical simplicity.

One condensed representation of itemsets is based on the concept of closure [1]. The closure $cl(X)$ of an itemset X is the maximal superset of the itemset X with the same support as the itemset X. The closure is always unique. An itemset X is a closed itemset if and only if its closure is the itemset X itself. We denote the collection of closed frequent itemsets by $Closed$.

Given $Freq$ we can obtain $Closed$ by taking the closure of each frequent itemset. Vice versa, given $Closed$ and an itemset, we obtain the support of the itemset by finding its most frequent superset.

A more recent proposal [2] for pruning redundant itemsets takes advantage of the inclusion-exclusion principle. If $Y \subset X$ and $|X \setminus Y|$ is odd, then the corresponding deduction rule for an upper bound of $s(X)$ is

$$s(X) \leq \sum_{I:Y \subset I \subset X, I \neq X} (-1)^{|X \setminus I|+1} s(I). \tag{1}$$

If $|X \setminus Y|$ is even, the direction of inequality is changed and the deduction rule gives a lower bound instead of an upper bound. Given all subsets of X, and their supports, we obtain a set of upper and lower bounds for X. When the smallest upper bound equals the highest lower bound, we have obtained the exact support of X. Such an itemset is called derivable. The collection of non-derivable frequent itemsets, denoted NDI, is a lossless representation of $Freq$. NDI is downward closed [2]. In other words, all supersets of a derivable itemset are derivable, and all subsets of a non-derivable itemset are non-derivable.

Example. Consider the small transaction dataset of four items a, b, c and d and six transactions given in the left panel of Figure 1. The deduction rules for computing bounds for $\{a, c, d\}$ are given in the right panel of Figure 1.

Example Dataset

a	b	c	d
		c	d
a	b		
a	b		d
a		c	
a		c	d

Deduction rules for $s(\{a, c, d\})$

$$
\begin{array}{ll}
s(\{a,c,d\}) \leq & s(\{a,c\}) + s(\{a,d\}) + s(\{c,d\}) \\
& -s(\{a\}) - s(\{c\}) - s(\{d\}) + s(\{\}) \\
& = 3 + 3 + 3 - 5 - 4 - 4 + 6 = 2 \\
s(\{a,c,d\}) \geq & s(\{a,c\}) + s(\{a,d\}) - s(\{a\}) = 3 + 3 - 5 = 1 \\
s(\{a,c,d\}) \geq & s(\{a,c\}) + s(\{c,d\}) - s(\{c\}) = 3 + 3 - 4 = 2 \\
s(\{a,c,d\}) \geq & s(\{a,d\}) + s(\{c,d\}) - s(\{d\}) = 3 + 3 - 4 = 2 \\
s(\{a,c,d\}) \leq & s(\{a,c\}) = 3 \\
s(\{a,c,d\}) \leq & s(\{a,d\}) = 3 \\
s(\{a,c,d\}) \leq & s(\{c,d\}) = 3 \\
s(\{a,c,d\}) \geq & 0
\end{array}
$$

Fig. 1. An example dataset and deduction rules

The support of the itemset $\{a, c, d\}$ has a highest lower bound of 2, which is equal to the lowest upper bound making $\{a, c, d\}$ derivable. □

3 Closed Non-derivable Itemsets

Both the collection of frequent itemsets and the collection of non-derivable itemsets are downward closed, and the use of closed itemsets for compression utilizes this property.

Definition 1. *Let NDI be the collection of frequent non-derivable itemsets. The collection of frequent closed non-derivable itemsets is $CNDI = \{cl(X) \mid X \in NDI\}$.*

There are some subtleties. First, $CNDI \subset NDI$ does not hold in general (whereas always $Closed \subset Freq$). From this it also follows that NDI cannot be reconstructed by a straightforward downward closure of $CNDI$.

Example. Given our example dataset in Figure 1 and a support threshold of 2, we have 12 frequent, 10 closed and 10 non-derivable itemsets, given in Figure 2.

Freq	*Closed*
{}(6)	{}(6)
{a}(5) {b}(3) {c}(4) {d}(4)	{a}(5) {c}(4) {d}(4)
{a,b}(3) {a,c}(3) {a,d}(3) {b,d}(2) {c,d}(3)	{a,b}(3) {a,c}(3) {a,d}(3) {c,d}(3)
{a,b,d}(2) {a,c,d}(2)	{a,b,d}(2) {a,c,d}(2)

NDI	*CNDI*
{}(6)	{}(6)
{a}(5) {b}(3) {c}(4) {d}(4)	{a}(5) {c}(4) {d}(4)
{a,b}(3) {a,c}(3) {a,d}(3) {b,d}(2) {c,d}(3)	{a,b}(3) {a,c}(3) {a,d}(3) {c,d}(3)
	{a,b,d}(2)

Fig. 2. All, closed, non-derivable and closed non-derivable itemsets and their supports

The collection of closed itemsets has been obtained by taking the closure of each frequent itemset. With regard to NDI, we showed earlier in Figure 1 that $\{a, c, d\}$ is a derivable itemset. This is also true for $\{a, b, d\}$.

As this example shows, neither NDI nor $Closed$ is a subset of the other. For instance, $\{a, b, d\}$ is closed but derivable whereas $\{b\}$ is non-derivable but not closed. On the other hand, $CNDI \subset Closed$. A comparison of these collections shows how $\{a, c, d\}$ is not in $CNDI$ since it is derivable. We emphasize that $CNDI$ cannot in general be obtained by removing all derivable itemsets from $Closed$. In this example, $|CNDI| = 9$, which is less than $|Closed| = 10$ and $|NDI| = 10$. □

Depending on the dataset, either collection of closed or collection of non-derivable itemsets may give a more condensed representation. However, the size of the collection of closed non-derivable itemsets is guaranteed to always be at most as large as the smallest of the two representations.

Theorem 1. *The size of the collection of closed non-derivable itemsets $CNDI$ is smaller than or equal to the size of the collection of non-derivable itemsets NDI: $|CNDI| \leq |NDI|$.*

Proof. By definition $CNDI = \{cl(X) \mid X \in NDI\}$. Since the closure operation gives exactly one itemset, we trivially have $|CNDI| \leq |NDI|$. □

Theorem 2. *The size of the collection of closed non-derivable itemsets $CNDI$ is smaller than or equal to the size of the collection of closed itemsets Closed: $|CNDI| \leq |Closed|$.*

Proof. By the definition of $CNDI$, given any $X \in CNDI$ there exists $Y \in NDI$ such that $X = cl(Y)$. Since $Y \in NDI \Rightarrow Y \in Freq \Rightarrow cl(Y) \in Closed \Rightarrow X \in Closed$, i.e., $CNDI \subset Closed$. □

Experiments in the following section show that the reduction by closed non-derivable itemsets may be significant in comparison to the reduction given by either of the methods alone.

The definition of $CNDI$ directly gives the simple Algorithm 1. Correctness of the algorithm is trivial and follows directly from the definition.

Algorithm 1. CloseNDI($\mathcal{D}$, δ)

Input: A dataset $\mathcal{D}$ and a support threshold δ
Output: The collection $CNDI$ of closed non-derivable itemsets

1: $CNDI \leftarrow \emptyset$
2: Mine all non-derivable itemsets NDI with support threshold δ from dataset $\mathcal{D}$ (using methods from [2,4]).
3: **for all** $X \in NDI$ **do**
4: $\quad CNDI \leftarrow CNDI \cup \{cl(X)\}$
5: **end for**
6: **return** $CNDI$

Obtaining all frequent itemsets and their supports from the closed non-derivable itemsets is a bit more complex operation. First, given a collection of closed non-derivable itemsets we can obtain the supports of all non-derivable itemsets. Then the non-derivable itemsets in turn can be used to deduce the supports of all frequent itemsets by a levelwise search of the itemset space.

Theorem 3. *Algorithm 2 correctly recovers the frequent itemsets and their supports from the collection of closed non-derivable frequent itemsets $CNDI$.*

Algorithm 2. ExpandCNDI($CNDI$, δ)

Input: A collection $CNDI$ of frequent closed non-derivable itemsets and the corresponding support threshold δ
Output: The collection $Freq$ of frequent itemsets

```
 1: Freq ← ∅
 2: l ← 0
 3: while ∃X such that |X| = l and for all Y ⊂ X, Y ≠ X: Y ∈ Freq do
 4:    for all X such that |X| = l and for all Y ⊂ X, Y ≠ X: Y ∈ Freq do
 5:       if support of X can be derived from supports of its proper subsets Y then
 6:          s ← Support of X as given by deduction rules (Equation 1)
 7:          if s ≥ δ then
 8:             Freq ← Freq ∪ (X, s)
 9:          end if
10:       else if ∃Z ∈ CNDI such that X ⊂ Z then
11:          s ← Support of the most frequent itemset Z ∈ CNDI for which X ⊂ Z
12:          Freq ← Freq ∪ (X, s)
13:       end if
14:    end for
15:    l ← l + 1
16: end while
17: return Freq
```

Proof. Let F be the true collection of all frequent itemsets while $Freq$ denotes the result of the algorithm. We first show that $F \subset Freq$ by induction over frequent itemsets $X \in F$ in increasing size

- For the empty set we have $s(\{\}) \geq \delta \Rightarrow \{\} \in NDI \Rightarrow cl(\{\}) \in CNDI$ by definition of $CNDI$. The algorithm can not derive the support of the empty set from its proper subsets (there are none). It thus proceeds to find the support of the closure of the empty set on line 11 of the algorithm and by properties of closed sets obtains the correct support.
- Let us now assume that the algorithm has correctly found all frequent itemsets with size less than l, and consider a frequent itemset $X \in F$ with $|X| = l$. Now $X \in F \Rightarrow X \in NDI$ or $X \in F \setminus NDI$. If $X \in F \setminus NDI$ then by definition of NDI its support can be derived from its proper subsets, which have been correctly recovered by algorithm (the inductive hypothesis). X and its support are thus correctly added to $Freq$ on line 8. If $X \in NDI$ then $cl(X) \in CNDI$ by definition of $CNDI$, and the algorithm finds the correct support of X from its closure on line 11.

To complete the proof we need to show that $Freq \subset F$. Consider any $X \in Freq$. It must have been added to $Freq$ on line 8 or 12. If it was added on line 8, $s(X) \geq \delta$ due to the test on line 7. If it was added on line 12, $s(X) \geq \delta$ since there exists $Z \in CNDI, X \subset Z$ (line 9) and by the definition of $CNDI$, Z must be frequent. □

The most time consuming phase of the Algorithm 1 is mining NDI. Time to close that collection depends on the number of itemsets in that collection. Algorithm 2

uses inclusion-exclusion to check whether an itemset is derivable. Again this is the dominant phase of the algorithm. For further analysis and efficient ways of implementing inclusion-exclusion we refer to [2,4].

4 Experiments

For an experimental evaluation of the proposed algorithms, we performed several experiments on real datasets. We implemented all algorithms in C++.

For primary comparison of methods we use four dense datasets: chess, connect, mushroom and pumsb, all obtained from the UCI Machine Learning Repository. The chess and connect datasets are derived from their respective game steps, the mushroom database contains characteristics of various species of mushrooms, and the pumsb dataset contains census data. For further perspective into the compression capabilities of the methods, we also use two sparse datasets, T10I4D100K and T40I10D100K, which contain simulated market basket data generated by the IBM Almaden Quest research group. Table 1 shows some characteristics of the used datasets.

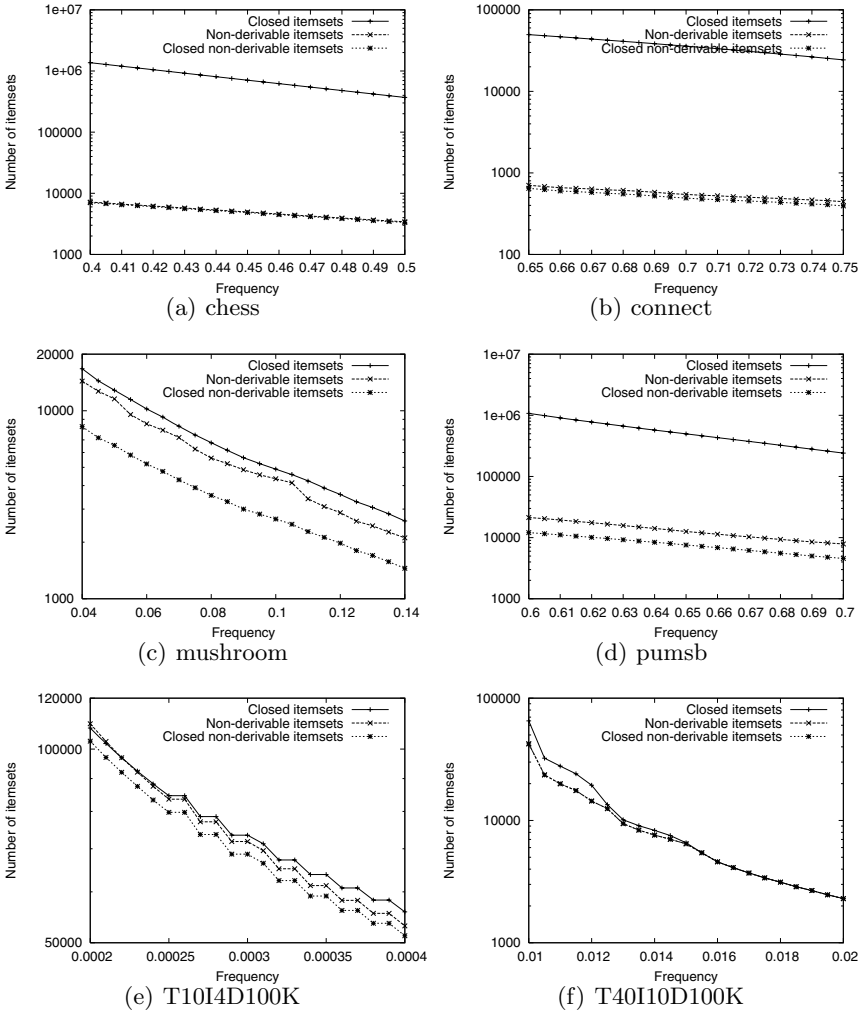
Table 1. Dataset characteristics

Dataset	Items	Avg. Trans. Size	Std. Dev. of Trans. Size	Transactions
chess	76	37	0	3 196
connect	130	43	0	67 557
mushroom	120	23	0	8 124
pumsb	7117	74	0	49 046
T10I4D100K	1000	10	3.7	100 000
T40I10D100K	1000	40	8.5	100 000

Figure 3 shows the results of the experiments: the number of closed itemsets, non-derivable itemsets and closed non-derivable itemsets, in the six different datasets as well as some of the results in numerical form.

The benefit of using closed non-derivable itemsets is biggest in mushroom and pumsb datasets, where the number of closed non-derivable itemsets is about one half of the number of non-derivable itemsets, and the reduction is even greater when compared to the number of closed itemsets (note that y-axis is logarithmic). In chess and connect datasets the compression for both non-derivable itemsets and closed non-derivable itemsets, when compared to closed itemsets, is about two orders of magnitude.

In all the dense datasets, the number of closed non-derivable itemsets is two to four magnitudes smaller than the number of all frequent itemsets. In the sparse market basket datasets, compression rates are much more modest, ranging about $15 - 35\%$. These two datasets are examples of the performance guarantee of closed non-derivable itemsets: for T10I4D100K, there are fewer closed itemsets than there are non-derivable itemsets, whereas in T40I10D100K the situation is reversed. In both cases the number of closed non-derivable itemsets is guaranteed not the exceed the smaller of these.



	chess	connect	mushroom	pumsb	T10	T40
Support Threshold (freq.)	1 279 (40%)	43 912 (65%)	325 (4.0%)	29 428 (60%)	20 (0.020%)	1 000 (1.0%)
$\|Freq\|$	6 472 981	9 727 092	5 131 852	19 537 366	129 876	65 237
$\|Closed\|$	1 366 834	49 707	16 733	1 075 015	107 823	65 237
$\|NDI\|$	7 185	704	14 382	21 323	109 486	42 312
$\|CNDI\|$	7 015	646	8 240	12 081	102 869	42 312

Fig. 3. The number of closed, non-derivable and closed non-derivable itemsets

5 Conclusions

We proposed a new method for lossless compression of a collection of frequent itemsets. The method takes advantage of the properties of two well-known techniques, closed itemsets and non-derivable itemsets.

We showed that the collection of closed non-derivable itemsets is a subset of the collection of closed itemsets, and that its size is also limited by the number of non-derivable itemsets, i.e., the combined method is guaranteed to yield better results than either one of the methods alone. We gave simple algorithms for producing the collection of closed non-derivable itemsets and recovering the collection of all frequent itemsets.

It is well known that closed itemsets and non-derivable itemsets give best compression rates for dense datasets, such as the UCI datasets used in our experiments, and give less benefits with sparse data, such as the IBM market basket data. Our experiments indicate that this is the case also for closed non-derivable itemsets. In our experiments with four real, dense datasets the reduction over closed itemsets was always significant ($50-99\%$). For two of them, the reduction over non-derivable itemsets was small, but for the two others the collection of closed non-derivable itemsets was approximately 43% smaller. This shows that the collection of closed non-derivable itemsets can in practice be significantly smaller than the two other condensed representations.

It is not easy to characterize datasets that compress particularly well with closed non-derivable itemsets. An obvious factor is density: dense datasets lend themselves better for compression. The advantage over closed and non-derivable itemsets is largest when their compressions are complementary, as the use of closed non-derivable itemsets can then benefit from both, as seemed to be the case with mushroom and pumsb datasets. More research is needed to better understand the different factors. On the other hand, for practical applications such understanding is not needed: regardless of the data, closed non-derivable itemsets are the optimal choice among the three compared alternatives.

References

1. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Proceedings of the 7th International Conference on Database Theory. Volume 1540 of Lecture Notes in Computer Science., Springer (1999) 398–416
2. Calders, T., Goethals, B.: Mining all non-derivable frequent itemsets. In: Proceedings of the sixth european conference on principles of data mining and knowledge discovery. (2002) 74–85
3. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD conference on management of data. (1993) 207–216
4. Calders, T., Goethals, B.: Quick inclusion-exclusion. In: KDID. (2005) 86–103

Learning a Distance Metric for Object Identification Without Human Supervision

Satoshi Oyama and Katsumi Tanaka

Department of Social Informatics, Graduate School of Informatics,
Kyoto University,
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
{oyama, tanaka}@dl.kuis.kyoto-u.ac.jp
http://www.dl.kuis.kyoto-u.ac.jp

Abstract. A method is described for learning a distance metric for use in object identification that does not require human supervision. It is based on two assumptions. One is that pairs of different names refer to different objects. The other is that names are arbitrary. These two assumptions justify using pairs of data items for objects with different names as "cannot-be-linked" example pairs for learning a distance metric for use in clustering ambiguous names. The metric learning is formulated using only dissimilar example pairs as a convex quadratic programming problem that can be solved much faster than a semi-definite programming problem, which generally must be solved to learn a distance metric matrix. Experiments on author identification using a bibliographic database showed that the learned metric improves identification F-measure.

1 Introduction

Object identification, which is used for example to determine whether the names of people in documents or databases refer to the same person or not, is an important problem in information retrieval and integration. It is most often used for personal name disambiguation, e.g., author identification in bibliographic databases. Citation and bibliographic databases are particularly troublesome because author first names are often abbreviated in citations. Resolving these ambiguities is necessary when evaluating the activity of researchers, but major citation databases such as the ISI Citation Index[1] and Citeseer's Most Cited Authors in Computer Science[2] cannot distinguish authors with the same first name initial and last name.

Object identification problems are generally solved by clustering data containing the target names based on some similarity measure or distance metric [1]. Similarity and distance are important factors in clustering, and an appropriate similarity/distance measure must be used to achieve accurate results. Several methods have been proposed for learning a similarity measure [2,3] or distance

[1] http://isiknowledge.com/
[2] http://citeseer.ist.psu.edu/mostcited.html

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 609–616, 2006.
© Springer-Verlag Berlin Heidelberg 2006

metric [4] from humanly labeled data. One advantage of using a distance metric rather than a general similarity/dissimilarity measure is that it satisfies mathematical properties such as the triangle inequality and can be used in many existing clustering algorithms. One problem in learning a distance metric is that labeling by a person involves costs. In previous research, labeling was given as pairwise feedback, such as two data items are similar and must be in the same cluster ("must-be-linked") or dissimilar and cannot be in the same cluster ("cannot-be-linked"), but disambiguating two people with the same name or similar names is a subtle and time-consuming task even for a person.

We have developed a distance metric learning method that requires no human supervision for object identification. It is based on two assumptions.

Different names refer to different objects. In many object identification problems, pairs of different names presumably refer to different objects with few exceptions. For example, two J. Smiths are ambiguous, while J. Smith and D. Johnson cannot be the same person (neglecting, of course, the possibility of false names or nicknames).

Names are arbitrary. There is no reason to believe that the data for two people with the same name are more similar than the data for two people with different names. For example, the research papers written by two different J. Smiths are not assumed to be more similar than those written by J. Smith and D. Johnson. We assume that a pair of data items for two people with different names has the same statistical properties as a pair of data items for two people with the same name.

These two assumptions justify the use of pairs of data items collected for different names (for example, J. Smith and D. Johnson) as cannot-be-linked examples for learning a distance metric to be used for clustering data for people with the same or similar names. The learned distance metric that gives good separation of the data for people with different names can be expected to separate the data for different people with the same name as well. These cannot-be-linked example pairs can be formed mechanically without manual labeling. In our setting, no similar (must-be-linked) example pairs are used. After formulating the distance metric learning problem with only dissimilar example pairs as a convex quadratic programming problem, we present experimental results for author identification using a bibliographic database.

2 Preliminaries

In this paper, $\boldsymbol{x}^m \in \mathcal{X}$ denotes data (documents or database records) that contain names, where the superscript m is the index for each data item. Each data item $\boldsymbol{x}^m$ is represented as a D dimensional feature vector $(x_1^m, \ldots, x_D^m)^T$, in which each feature corresponds to, for example, a word in a document or an attribute in a database. The superscript T denotes the transpose of a vector or matrix.

Given vector representations of the data, we can define various distance metrics. For the function $d : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ to be a (pseudo) metric, it must satisfy the following conditions[3]:

$$\begin{aligned} &d(\boldsymbol{x}^m, \boldsymbol{x}^n) \geq 0 \\ &d(\boldsymbol{x}^m, \boldsymbol{x}^n) = d(\boldsymbol{x}^n, \boldsymbol{x}^m) \\ &d(\boldsymbol{x}^m, \boldsymbol{x}^l) + d(\boldsymbol{x}^l, \boldsymbol{x}^n) \geq d(\boldsymbol{x}^m, \boldsymbol{x}^n) \ . \end{aligned}$$

The Euclidean metric treats each feature equally and independently and does not represent interaction among features. Using $D \times D$ matrix $\boldsymbol{A} = \{a_{i,j}\}$, we can define a distance metric in a more general form:

$$\begin{aligned} d_{\boldsymbol{A}}(\boldsymbol{x}^m, \boldsymbol{x}^n) &= \left((\boldsymbol{x}^m - \boldsymbol{x}^n)^T \boldsymbol{A} (\boldsymbol{x}^m - \boldsymbol{x}^n)\right)^{\frac{1}{2}} \\ &= \left(\sum_{i=1}^{D} \sum_{j=1}^{D} a_{i,j} (x_i^m - x_i^n)(x_j^m - x_j^n) \right)^{\frac{1}{2}} \ . \end{aligned}$$

The necessary and sufficient condition for $d_{\boldsymbol{A}}$ being a pseudo metric is that $\boldsymbol{A}$ be a positive semi-definite matrix, in other words, a symmetric matrix in which all eigenvalues are non negative. Xing *et al.* [4] proposed a distance metric learning method in which similar and dissimilar pairs of examples are given, and a matrix $\boldsymbol{A}$ is found that minimizes the sum of the distances between similar pairs while keeping the distances between dissimilar pairs greater than a certain value. However, the optimization problem includes a constraint: matrix $\boldsymbol{A}$ must be positive semi-definite. We thus have a semi-definite programming problem [5], which is harder to solve than a convex quadratic programming problem, like that used in support vector machine learning [6].

3 Distance Metric Learning from Only Dissimilar Example Pairs

3.1 Problem Formalization

In our setting, only pairs of dissimilar (cannot-be-linked) examples $(\boldsymbol{x}^m, \boldsymbol{x}^n) \in \mathcal{D}$ are given, where $\mathcal{D} \subset \mathcal{X} \times \mathcal{X}$ is the set of paired examples that are considered to be referring to different objects, that is, examples with different names.

We want examples in such a pair to belong to different clusters. To ensure that, we use a matrix $\boldsymbol{A}$ that enlarges the distance between the two examples $d_{\boldsymbol{A}}(\boldsymbol{x}^m, \boldsymbol{x}^n)$. However, multiplying $\boldsymbol{A}$ by a large scalar makes the distance between any two points long and thus not meaningful. Therefore, we introduce a constraint that the norm of matrix $\boldsymbol{A}$ must be a certain constant, say 1, and find the $\boldsymbol{A}$ that induces a long distance between dissimilar examples in a pair while

[3] d becomes a metric in the strict sense when $d(\boldsymbol{x}^m, \boldsymbol{x}^n) = 0$ if and only if $\boldsymbol{x}^m = \boldsymbol{x}^n$.

satisfying the constraint. As the matrix norm, we use the Frobenius norm:

$$\|\boldsymbol{A}\|_F = \left(\sum_{i=1}^{D} \sum_{j=1}^{D} a_{i,j}^2 \right)^{\frac{1}{2}} .$$

We can now formalize distance metric learning from only dissimilar example pairs as an optimization problem:

$$\max_{\boldsymbol{A}} \min_{(\boldsymbol{x}^m, \boldsymbol{x}^n) \in \mathcal{D}} \quad d_{\boldsymbol{A}}(\boldsymbol{x}^m, \boldsymbol{x}^n) \tag{1}$$

$$\text{s.t.} \quad \|\boldsymbol{A}\|_F = 1 \tag{2}$$

$$\boldsymbol{A} \succeq 0 \ . \tag{3}$$

$\boldsymbol{A} \succeq 0$ means that $\boldsymbol{A}$ should be positive semi-definite. Objective function (1) requires finding the $\boldsymbol{A}$ that maximize the distance between the closest example pair. This idea is similar to large margin principles in SVMs [6] and is justified because clustering errors most probably occur at the cannot-be-linked points closest to each other, and keeping these points far from each other reduces the risk of errors.

To simplify the subsequent calculation, we translate the above problem into an equivalent one:

$$\min_{\boldsymbol{A}} \quad \frac{1}{2}\|\boldsymbol{A}\|_F^2 \tag{4}$$

$$\text{s.t.} \quad d_{\boldsymbol{A}}(\boldsymbol{x}^m, \boldsymbol{x}^n) \geq 1 \quad \forall (\boldsymbol{x}^m, \boldsymbol{x}^n) \in \mathcal{D} \tag{5}$$

$$\boldsymbol{A} \succeq 0 \ . \tag{6}$$

3.2 Positive Semi-definiteness of Learned Matrix

We now consider an optimization problem consisting of only (4) and (5) without (6). To solve this problem, we introduce the Lagrangean

$$\begin{aligned} L(\boldsymbol{A}, \boldsymbol{\alpha}) =& \frac{1}{2}\|\boldsymbol{A}\|_F^2 + \sum_{(m,n)} \alpha^{(m,n)} \left(1 - d_{\boldsymbol{A}}(\boldsymbol{x}^m, \boldsymbol{x}^n)\right) \\ =& \frac{1}{2}\|\boldsymbol{A}\|_F^2 + \sum_{(m,n)} \alpha^{(m,n)} \left(1 - (\boldsymbol{x}^m - \boldsymbol{x}^n)^T \boldsymbol{A} (\boldsymbol{x}^m - \boldsymbol{x}^n)\right) , \end{aligned} \tag{7}$$

with Lagrange multipliers $\alpha^{(m,n)} \geq 0$.

In the solution of (4) and (5), the derivative of $L(\boldsymbol{A}, \boldsymbol{\alpha})$ with respect to $\boldsymbol{A}$ must vanish; that is, $\frac{\partial L}{\partial \boldsymbol{A}} = 0$. This leads to the following expansion:

$$\boldsymbol{A} = \sum_{(m,n)} \alpha^{(m,n)} (\boldsymbol{x}^m - \boldsymbol{x}^n)(\boldsymbol{x}^m - \boldsymbol{x}^n)^T \ . \tag{8}$$

A necessary and sufficient condition for $D \times D$ matrix $\boldsymbol{A}$ being positive semi-definite is that for all D dimensional vectors $\boldsymbol{v}$, $\boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} \geq 0$ holds. This is always

the case for a matrix $\boldsymbol{A}$ in the form of (8). Noting that $\alpha^{(m,n)} \geq 0$, we can confirm this as follows:

$$\boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} = \sum_{(m,n)} \alpha^{(m,n)} ((\boldsymbol{x}^m - \boldsymbol{x}^n)^T \boldsymbol{v})^2 \geq 0 \ .$$

This means that without condition (6), the positive semi-definiteness of $\boldsymbol{A}$ is automatically satisfied. In fact, the optimization problem consisting of only (4) and (5) is a convex quadratic programming problem and can be solved much faster than a semi-definite programming problem with condition (6).

3.3 Relationship to Support Vector Machine Learning

Our formalization of learning a distance metric from only dissimilar example pairs is closely related to support vector machine learning. Actually, the optimization problem can be translated into an SVM learning problem [6] and can be solved by existing SVM software with certain settings.

The optimization problem for training an SVM that classifies the data into two classes is as follows [6]:

$$\min_{\boldsymbol{w},b} \ \frac{1}{2} \|\boldsymbol{w}\|^2 \tag{9}$$

$$\text{s.t.} \ \ y^m(\langle \boldsymbol{w}, \boldsymbol{x}^m \rangle + b) \geq 1 \ \ \forall (\boldsymbol{x}^m, y^m) \in \mathcal{T} \ . \tag{10}$$

$\mathcal{T}$ is the set of training examples $(\boldsymbol{x}^m, y^m)$, where $\boldsymbol{x}^m$ is a data vector and $y^m \in \{-1, +1\}$ is the class label. $\langle \boldsymbol{x}, \boldsymbol{z} \rangle$ is the inner product of vectors $\boldsymbol{x}$ and $\boldsymbol{z}$.

Using the Frobenius product

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle_F = \sum_{i=1}^{D} \sum_{j=1}^{D} a_{i,j} b_{i,j}$$

of two $D \times D$ matrices, we can rewrite the problem of (4) and (5):

$$\min_{\boldsymbol{A}} \ \frac{1}{2} \|\boldsymbol{A}\|_F^2 \tag{11}$$

$$\text{s.t.} \ \ \langle \boldsymbol{A}, (\boldsymbol{x}^m - \boldsymbol{x}^n)(\boldsymbol{x}^m - \boldsymbol{x}^n)^T \rangle_F \geq 1 \ \ \forall (\boldsymbol{x}^m, \boldsymbol{x}^n) \in \mathcal{D} \ . \tag{12}$$

Comparison of (11) and (12) with (9) and (10) reveals that our problem corresponds to unbiased SVM learning ($b = 0$) from only positive data ($y^m = 1$), if we consider the examples and the learned weight of $D \times D$ matrices as D^2 dimensional vectors. The expansion form of the SVM solution $\boldsymbol{w} = \sum_m y^m \alpha^m \boldsymbol{x}^m$ makes clear why our method can avoid semi-definite programming. We use only positive examples (cannot-be-linked pairs), thus all the coefficients for the examples become positive in the solution. If we also used negative examples (must-be-linked pairs), the coefficients for these examples become negative and the solution is not always positive semi-definite.

Substituting (8) into (7) gives us the dual form of the problem:

$$\begin{aligned} \max \quad & \sum_{(m,n)} \alpha^{(m,n)} \\ & -\frac{1}{2} \sum_{(m,n)} \sum_{(m',n')} \left(\alpha^{(m,n)} \alpha^{(m',n')} \langle \boldsymbol{x}^m - \boldsymbol{x}^n, \boldsymbol{x}^{m'} - \boldsymbol{x}^{n'} \rangle^2 \right) \\ \text{s.t.} \quad & \alpha^{(m,n)} \geq 0 \ . \end{aligned}$$

These formulas indicate that our learning problem can be solved by using the quadratic polynomial kernel on D dimensional vectors and that we do not need to calculate the Frobenius products between the $D \times D$ matrices. As with standard SVMs, our method can be "kernelized" [7]. By substituting a positive semi-definite kernel function $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{z}) \rangle$ ($\boldsymbol{\phi}(\boldsymbol{x})$ is a map to a higher dimensional space) for the inner product $\langle \boldsymbol{x}, \boldsymbol{z} \rangle$, we can virtually learn the distance metric matrix for a very high (possibly infinite) dimensional feature space by the so-called "kernel trick." In addition, a distance metric for structured data, such as trees or graphs, can be learned with a kernel function defined on the space of such data.

4 Experiments

We tested our method on the DBLP data set, which is a bibliography of computer science papers.[4] The entries were made by people, and many author names include the full first name, not only an initial. We assume that the same first and last names refers to the same person.

From among the Most Cited Authors in Computer Science,[5] we selected eight cases of first-initial-plus-surname names, which involve a collapsing of many distinct author names. We retrieved papers written by authors with the same last name and the same first initial from the DBLP data and randomly selected 100 examples for each abbreviated name. Then we abbreviated first names into initials and removed middle names. Training data were built by pairing examples of different abbreviated names, for example, J. Smith and D. Johnson. We used words in titles, journal names, and names of coauthors as features. Since few words appear more than once in a bibliographic entry, we used binary features.

To learn a distance metric, we used SVM^{light}[8]. The learned metric was used in clustering the data from the same-first-initial-and-last authors. We used the single-linkage clustering algorithm [9]. The results of clustering were evaluated by referring to the original full names.

The results with the learned metric were compared to the results with two other metrics, one was the Euclidean distance and the other was the IDF weighting [10]. Since each bibliography entry is short and the same word rarely appears

[4] http://dblp.uni-trier.de/

[5] http://citeseer.ist.psu.edu/mostcited.html

Table 1. Maximum F-measure values

Abbreviated name	F-measure			Abbreviated name	F-measure		
	Learned	IDF	Euclidean		Learned	IDF	Euclidean
D. Johnson	.644	.390	.399	L. Zhang	.278	.165	.158
A. Gupta	.490	.170	.169	H. Zhang	.423	.226	.226
J. Smith	.417	.270	.292	R. Jain	.709	.569	.552
R. Johnson	.508	.253	.227	J. Mitchell	.640	.535	.536

more than once in the entry, we did not apply TF weighting. We neither normalized the feature vectors because the lengths of bibliographic entries are rather uniform. The clustering algorithm enables us to specify the number of clusters. We measured the pairwise precision and recall for each number of clusters. The maximum F-measure (harmonic mean of precision and recall [10]) for each combination of name and metric is given in Table 1. Use of the learned metric consistently resulted in the highest F-measure, while the values varied for different names.

5 Related and Future Work

Xing *et al.* [4] proposed a distance metric learning from similar and dissimilar example pairs. They formulated the problem as a semi-definite programming problem, and their algorithm needs a full eigenvalue decomposition to ensure that the learned matrix is positive semi-define. Schultz & Joachims [11] proposed a method for learning a distance metric from relative comparison such as "A is closer to B than A is to C." They also formulated the metric learning as a constrained quadratic programming. In their method, the interactions between features are fixed and optimization is applied to a diagonal matrix. Our method can learn a full distance metric matrix by using only cannot-be-linked pairs.

Shalev-Shwartz, Singer & Ng [12] proposed an online learning algorithm for learning a distance metric. Their algorithm does not strictly solve the constrained optimization problem; it finds successive approximate solutions using an iterative procedure that combines a perceptron-like update rule and the Lanczos method to find a negative eigenvalue. While designed for learning from both similar and dissimilar pairs, their algorithm can avoid the eigenvalue problem, as ours does, if it uses only dissimilar example pairs. The performance of the online kernel perceptron algorithm is close to, but not as good as, that of SVMs for the same problem, while saving significantly on computation time [13]. This suggests an interesting direction for future work: adopting online algorithms that learn only from dissimilar examples and comparing the results to those of our learning method.

6 Conclusion

We proposed a method for learning a distance metric for use in object identification that is based on two assumptions: different names refer to different objects

and the data for two people with exactly the same name are no more similar than the data for two people with different names. It learns the distance metric from only dissimilar example pairs, which are mechanically collected without human supervision. We formalized our learning problem as a convex quadratic programming problem, which can be efficiently solved by existing SVM software. Experiments using the DBLP data set showed that the learned metric improves F-measure for object identification.

Acknowledgements

This work was supported in part by a Grant-in-Aid for Scientific Research (No. 16700097) from MEXT of Japan, by a MEXT project titled "Software Technologies for Search and Integration across Heterogeneous-Media Archives," and by a 21st Century COE Program at Kyoto University titled "Informatics Research Center for Development of Knowledge Society Infrastructure."

References

1. Mann, G.S., Yarowsky, D.: Unsupervised personal name disambiguation. In: Proc. CoNLL-2003. (2003) 33–40
2. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proc. KDD-2003. (2003) 39–48
3. Oyama, S., Manning, C.D.: Using feature conjunctions across examples for learning pairwise classifiers. In: Proc. ECML-2004. (2004) 322–333
4. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.J.: Distance metric learning, with application to clustering with side-information. In: Proc. NIPS-15. (2003) 505–512
5. Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM Review **38**(1) (1996) 49–95
6. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons (1998)
7. Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. MIT Press (2002)
8. Joachims, T.: Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods : Support Vector Learning. MIT Press (1999) 169–184
9. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall (1988)
10. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley (1999)
11. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: Proc. NIPS-16. (2004) 41–48
12. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Proc. ICML-2004. (2004)
13. Freund, Y., Schapire, R.E.: Large margin classification using the perceptron algorithm. Machine Learning **37**(3) (1999) 277–296

Towards Association Rules with Hidden Variables

Ricardo Silva[1] and Richard Scheines[2]

[1] Gatsby Unit, University College London,
WC1N 3AR London, UK
[2] Machine Learning Department, Carnegie Mellon,
Pittsburgh PA, 15213, USA

Abstract. The mining of association rules can provide relevant and novel information to the data analyst. However, current techniques do not take into account that the observed associations may arise from variables that are unrecorded in the database. For instance, the pattern of answers in a large marketing survey might be better explained by a few latent traits of the population than by direct association among measured items. Techniques for mining association rules with hidden variables are still largely unexplored. This paper provides a sound methodology for finding association rules of the type $H \Rightarrow A_1, \ldots, A_k$, where H is a hidden variable inferred to exist by making suitable assumptions and $A_1, \ldots, A_k$ are discrete binary or ordinal variables in the database.

1 Contribution

Consider the problem of discovering association rules of the type

$$H \Rightarrow A_1, A_2, \ldots, A_k \tag{1}$$

where H is a variable that is not present in the database (a *hidden*, or *latent* variable) but that explains the association among recorded discrete variables $\{A_1, \ldots, A_k\} \subseteq \{X_1, X_2, \ldots, X_N\}$. This paper provides a novel algorithm for mining such rules.

The motivation is two-fold: first, the outcome of such an analysis can aid the discovery of plausible and novel hidden variables that may be used to characterize the population of interest. Second, it might provide a more concise set of rules.

For instance, suppose that our data was generated by the graphical model shown in Figure 1, where H is hidden and $X_1, \ldots, X_4$ are observable. A typical association rule algorithm might find all sorts of rules such as $X_1 \Rightarrow X_2, X_3$, $X_2 \Rightarrow X_1, X_3, X_4$, etc. A hidden variable approach could in principle output a single rule subsumming all of such rules.

This paper is organized as follows: in Section 2, we introduce the particular class of hidden variable association rules we use, making the link to related work in latent variable graphical models. Section 3 is the main section of the paper, describing the detailed approach. Experiments are discussed in Section 4.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 617–624, 2006.
© Springer-Verlag Berlin Heidelberg 2006

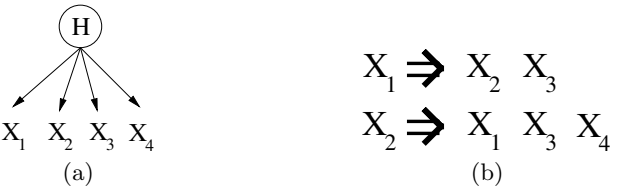


Fig. 1. Assume the data was generated according to the latent variable model in (a), where H is not recorded in the database and not known to exist. A potential set of association rules that ignore hidden variables is given in (b).

2 Probabilistic Formulation

Following the framework of [7], we assume that our data was generated by a causal *directed acyclic graph*, where an edge $A \rightarrow B$ has the meaning that "A is a direct cause of B". There are several advantages on trying to extract *subgraphs* of the original graph as a type of association rule, instead of discovering a full graph [7], as further discussed in Section 2.1. Assuming there is a true graph G that generates our data, the semantics of a latent association rule $H \Rightarrow A_1, \ldots, A_k$, as given in this paper, are:

- H is a hidden node and a common ancestor of $A_1, \ldots, A_k$ in G, i.e., H is a hidden common cause of all elements in $A_1, \ldots, A_k$
- all variables $A_1, \ldots, A_k$ are probabilistically dependent, but become independent when conditioning on H[1];

The problem of discovering hidden variables is ill-defined without making extra assumptions. That is, if H is allowed to assume any distribution, then any distribution over observed variables can be generated by a hidden variable. This can be accomplished, for instance, by making H be a discrete variable with as many states as the entries of the contingency table of $A_1 \times \ldots \times A_k$. Such a rule can never be falsified. Instead, we will assume that our data was generated by some model from the family of *latent trait models* [1].

A model of this class is more easily understood through a graphical representation, as illustrated in Figure 2(a): each directed edge $H \rightarrow X_i$ from hidden node H to observed node X_i can be interpreted by having some intermediate hidden variable X_i^* on the path, as in $H \rightarrow X_i^* \rightarrow X_i$. The underlying X_i^* with latent parents $\{H_1^{X_i}, \ldots, H_k^{X_i}\}$ is given by

$$X_i^* = \sum_{j=1}^{k} \lambda_{ij} H_j^{X_i} + \epsilon_i; \epsilon_i \sim \mathrm{N}(0, \sigma_i^2);$$

and each λ_{ij} corresponds to the linear effect of parent $H_j^{X_i}$ on X_i^*. Latent variables are assumed to follow a multivariate Gaussian distribution, centered at

[1] That is, unlike traditional association rules, the right-hand side of the rule is not meant to assume any particular value (e.g., $A_1 = true$). Instead, the interpretation is that $A_1, \ldots, A_k$ are associated, but independent conditioned on H.

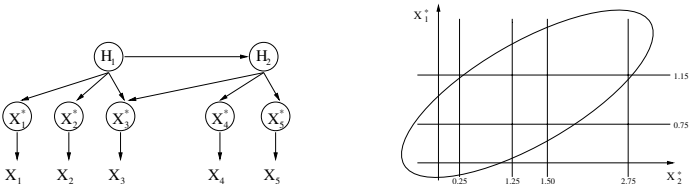


Fig. 2. (a) A graphical representation of a latent trait model with 5 ordinal observed variables. (b) Two ordinal variables X_1 and X_2 can be seen as discretizations of two continuous variables X_1 and X_2. The lines in the graph above represent thresholds that define the discretization. The ellipse represents a contourplot of the joint Gaussian distribution of the two underlying continuous variables X_1, X_2.

zero. The observed variables X_i are then just discretizations of the respective X_i^*, as illustrated in Figure 2(b). More details on this model are given by [1] and [5].

This model imposes *constraints* on the observed joint distribution of the ordinal variables. Different graphical structures imply different correlations, and this can be used to test plausible association rules, as discussed in Section 3.

Even though this family of models rely on strong parametric assumptions, it has been successfully used to model binary and ordinal data, particularly *survey data* such as marketing questionnaires, social sciences and public opinion polls. It is also the basis of several psychological and educational testing studies [1].

2.1 Related Work

What do we gain by extracting association rules from a graphical model instead of trying to learn the graphical structure directly? One major reason is *scalability*, as motivated by [7]: the data might have been generated by a directed graph that is too large to be efficiently learned from data. This is even more problematic in latent trait models, which requires the computation of high-dimensional Gaussian integrals. This scalability problem is also connected to the *statistical* problem of trying to learn large structures: different substructures of the graph might be more strongly supported by the data, and it would be of interest to report only on those substructures (i.e., association rules) of high confidence. Another major motivation is *identifiability*. As discussed at length by [6], there might be many different graphical structures that equally explain the data, but that agree on particular substructures. Rule mining focuses directly on those substructures that are uniquely identifiable from the assumptions.

Although there are other approaches for discovering latent variable models for discrete data (e.g., [3]), they do not address the issues raised above. The goal is usually density estimation, not knowledge discovery. Morever, they often assume that latent variables are marginally independent, an unnecessary assumption made mostly for the sake of simplicity.

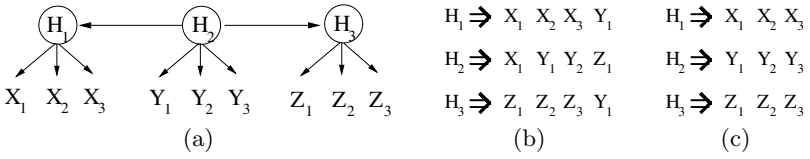


Fig. 3. If the data is generated by the structure in (a), Lemma 1 alone could be used to generate the rules in (b), with some incorrect information concerning the right-hand sides. However, the rules shown in (c) can be obtained by application of Lemma 2.

3 Methodology

We now describe an algorithm that generates rules corresponding to subgraphs of the (unknown) true graph G, which is assumed to have generated the data. Traditionally, conditional independency constraints are used to discover graphical structures. However, in a latent trait model, few, if any, of such constraints are observable [6]. Other types of constraints should be used. Consider a set of four variables, $\{W, X, Y, Z\}$ such that $\sigma_{WX}\sigma_{YZ} = \sigma_{WY}\sigma_{XZ} = \sigma_{WZ}\sigma_{XY}$, where σ_{XY} is the covariance of random variables X and Y. Under assumptions common in structure learning algorithms, the following holds in linear models [6]:

Lemma 1. *Let G be a linear latent variable model, and let $\{X_1, X_2, X_3, X_4\}$ be such that $\sigma_{X_1X_2}\sigma_{X_3X_4} = \sigma_{X_1X_3}\sigma_{X_2X_4} = \sigma_{X_1X_4}\sigma_{X_2X_3}$. If $\sigma_{AB} \neq 0$ for all $\{A, B\} \subset \{X_1, X_2, X_3, X_4\}$, then there is a node P conditioned on which all elements in $\{X_1, X_2, X_3, X_4\}$ are independent.*

This holds even if P is not observed, which means we can detect the existence of latent variables by using the covariance matrix of the given observed variables[2]. Lemma 1, however, does not provide enough information, since it does not indicate if such variables are descendants of P or not. To solve this issue, we rely on the following result (also in [6]):

Lemma 2. *If constraints $\sigma_{X_1Y_1}\sigma_{X_2X_3} = \sigma_{X_1X_2}\sigma_{X_3Y_1} = \sigma_{X_1X_3}\sigma_{X_2Y_1}$, $\sigma_{X_1Y_1}\sigma_{Y_2Y_3} = \sigma_{X_1Y_2}\sigma_{Y_1Y_3} = \sigma_{X_1Y_3}\sigma_{Y_1Y_2}$, $\sigma_{X_1X_2}\sigma_{Y_1Y_2} \neq \sigma_{X_1Y_1}\sigma_{X_2Y_2}$ all hold, then X_1 and Y_1 do not have a common parent in G.*

Notice that this result could be used to correct the rules in the example of Figure 3: one can verify that the above result holds for the pairs $\{X_1, X_2, X_3\} \times \{Y_1, Y_2, Y_3\}$, $\{Y_1, Y_2, Y_3\} \times \{Z_1, Z_2, Z_3\}$ and $\{X_1, X_2, X_3\} \times \{Z_1, Z_2, Z_3\}$.

What follows is an adaption of the algorithm in [6] to generate association rules. The main algorithm, BuildLatentRules (Table 1), starts by generating sets of variables (cliques) that could not be judged to measure different latents

[2] In our case variables are ordinal or binary, not continuous. However, there is an equivalent notion of covariance matrix for ordinal and binary variables, and tests of statistical significance for such constraints [5]. If there is enough memory to cache all second moments of the data, then this requires a single pass through the data.

Table 1. An algorithm for learning association rules with hidden variables

Algorithm BuildLatentRules
Input: dataset $\mathcal{D}$ with observed variables $\mathbf{X}$

1. Let C be a fully connected undirected graph with nodes in $\mathbf{X}$
2. Remove edge $X_i - X_j$ from C if X_i and X_j are statistically marginally independent
3. Remove $X_i - X_j$ if X_i and X_j can be statistically separated as in Lemma 2
4. Let $\mathbf{M}$ be the set of maximal cliques in C.
5. $\mathbf{R_C} \leftarrow$ PurifyIndividualSets$(\mathcal{D}, \mathbf{M})$.
6. Return FilterRedundant$(\mathbf{R_C})$.

Table 2. Identifying association rules from potential clusters of variables

Algorithm PurifyIndividualSets
Inputs: dataset $\mathcal{D}$ with observed variables $\mathbf{X}$
Sets, a set of subsets of $\mathbf{X}$;

1. **Output** $\leftarrow \emptyset$
2. Repeat Step 3 below for all $Set \in$ **Sets**
3. If there is some $\{W, X, Y, Z\} \subset Set$ such that constraint $\sigma_{XY}\sigma_{WZ} = \sigma_{XW}\sigma_{YZ} = \sigma_{XZ}\sigma_{WY}$ is not true according to a statistical test, remove the node in Set that participates in the largest number of violated constraints. Repeat until all constraints are satisfied.
4. If Set has at least three variables, add it to **Output**.
5. Return **Output**.

(using Lemma 2). However, failure to be separated by Lemma 2 does not imply such nodes indeed have latent parents in common. A second pass through such sets has to be performed to "purify" each set, resulting in a desirable association rule. This is performed by algorithm PurifyIndividualSets (Table 2): it ensures that Lemma 1 holds for any foursome in a selected set[3].

Because there might be several ways of "purifying" each candidate set, there might be many rules that are a consequence of the same hidden variable. Optionally, we might want to present just one rule for each hidden variable. This is performed by algorithm FilterRedundant defined as follows: if two rules overlap in three or more observed variables, then by Lemma 1 the hidden variable responsable for this pattern should be the same. FilterRedundant will allow only one rule for each hidden variable and also remove any rule whose right-hand size is contained in the union of other rules. This helps to minimize the number of spurious rules that are included by statistical mistakes.

[3] This algorithm requires a rule to have at least three variables on its right-hand side. For rules with fewer than three variables, see the complete algorithm in [5]. Moreover, for technical reasons omitted for lack of space, due to identifiability limitations of latent trait models it is possible that one (and at most one) of the elements on the right-hand side might actually not be a child of the latent (see [5,6]).

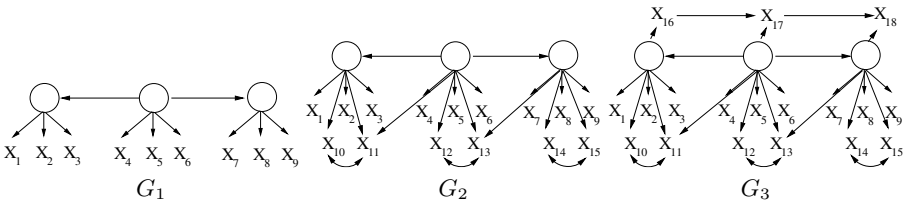


Fig. 4. The graphs used in our simulation studies

Table 3. Results obtained with BUILDLATENTRULES and APRIORI for the problem of learning latent rules. For BUILDLATENTRULES, each number is an average over 10 trials, with the standard deviation over these trials in parenthesis.

BUILDLATENTRULES **statistics**				
	Sample	**Precision**	**Recall**	**#Rules**
G_1	*1000*	1.00(.0)	0.97(.1)	3.2(.4)
	5000	0.98(.05)	0.97(.1)	2.9(.3)
G_2	*1000*	0.94(.04)	1.00(.0)	3.2(1.03)
	5000	0.94(.05)	1.00(.0)	3.4(0.70)
G_3	*1000*	0.90(.06)	0.90(.16)	4.2(.91)
	5000	0.90(.08)	0.90(.22)	3.5(.52)

APRIORI **statistics**					
	Sample	**MIN**	**MAX**	**AVG**	**STD**
G_1	*1000*	15	159	81	59.4
	5000	9	546	116	163.9
G_2	*1000*	243	2134	1070.4	681.2
	5000	336	3565	1554.7	1072.2
G_3	*1000*	363	6036	2916.7	1968.7
	5000	158	4434	2608.3	1214.6

4 Experiments

In the following sections we evaluate our algorithm in a series of simulated experiments, and exploratory results on a real data set. In the simulated cases, we report statistics about the number of association rules that the standard algorithm APRIORI (using the implementation of [2]) returns on the same data. The goal is to provide evidence that standard algorithms might produce thousands of rules, despite the simple underlying latent variable model.

For the simulation study, let G be our true graph, from which we want to extract association rules. The graph is known to us by simulation, but it is not known to the algorithm. The goal of experiments with synthetic data is to objectively measure the performance of BUILDLATENTRULES[4] in finding correct and informative latent rules. Correctness is measured by a *Precision* statistic: the average precision of each rule. The precision of a rule is the proportion of items on the right-hand size that are in fact independent given the latent on the left. Completeness is measured by a *Recall* statistic: the proportion of latents $\{H_i\}$ in G such that there is at least one corresponding rule in our output. In our study we use the three graphs depicted in Figure 4, where all latents are potentially identifiable. Given each graph, we generated 10 parametric models and a sample of size 1,000 from each. Other 10 models were generated to sample datasets of 5,000 cases. The sampling scheme is given in [5]. Results are shown

[4] We use a slightly different variation of the algorithm to preprocess feasible candidate rules. Details in [5], Chapter 5.

Table 4. Examples of rules obtained by BUILDLATENTRULES on Deck 6 of the Freedom and Tolerance data set (question number and respective textual description)

Rule 1	
X27	I feel it is more important to be sympathetic and understanding of other people than to be practical and tough-minded
X3	I like to discuss my experiences and feelings openly with friends instead of keeping them to myself
X31	People find it easy to come to me for help, sympathy, and warm understanding
X67	When I have to meet a group of strangers, I am more shy than most people
X7	I would like to have warm and close friends with me most of the time
Rule 2	
X28	I lose my temper more quickly than most people
X30	I often react so strongly to unexpected news that I say or do things I regret
X41	I often push myself to the point of exhaustion or try to do more than I can
X61	I find it upsetting when other people don't give me the support that I expect
Rule 3	
X9	I usually demand very good practical reasons before I am willing to change my old ways of doing things
X53	I see no point in continuing to work on something unless there is a good chance of success
X46	I like to think about things for a long time before I make a decision
Rule 4	
X3	I like to discuss my experiences and feelings openly with friends instead of keeping them to myself
X40	I am slower than most people to get excited about new ideas and activities
X12	My friends find it hard to know my feelings because I seldom tell them about my private thoughts

in Table 3. We also display the number of rules that are generated. Ideally, in all cases we should generate exactly 3 rules. Due to statistical mistakes, more or less than 3 rules can be generated. It is noticeable that there is a tendency to produce more rules than necessary as the graph increases in complexity. It is also worthy to point out that without the FILTERREDUNDANT algorithm, we obtain around around 5 to 8 rules in most of the experiments. As a comparison, we report the distribution of rules generated by APRIORI in Table 3. We report the maximum and minimum number of rules for each model and sample size across the 10 trials, as well as average and standard deviation. The outcome is that not only APRIORI generates a very large number of rules, but the actual number per trial varies enormously (see, e.g., G_1 at sample size 5000).

We also applied BUILDLATENTRULES to the data collected in a 1987 study[5] on freedom and tolerance in the United States [4]. This is a large study comprising 381 questions targeting political tolerance and perceptions of personal freedom in the United States. 1267 respondents completed the interview. Each question

[5] Available at http://webapp.icpsr.umich.edu/cocoon/ICPSR-STUDY/09454.xml

is an ordinal variable with 2 to 5 levels, often with an extra non-ordinal value corresponding to a "Don't know/No answer" reply. However, several questions are explicitly dependent on answers given to previous questions. To simplify the task, in this empirical evaluation we will focus on a particular section of this questionnaire, the Deck 6. This deck of questions is composed of a self-administred questionnaire of 69 items concerning an individual's attitude with respect to other people. We obtained 15 rules, where 40 out of the 69 questions appear on at least on rule. Some of such rules are depicted in Table 4. There is a clear relation among items within most rules. For instance, items on Rule 1 correspond to measures of a latent trait of empathy and easiness of communication. Rule 2 has three items (X28, X30, X61) that clearly correspond to measures of a tendency of impulsive reaction. The fourth item (X41) is not clearly related to this trait, but the data supports the idea that this latent trait explains the associations between pushing oneself too much and reacting strongly to other people. See [5] for a more extensive discussion.

5 Conclusion

Our approach should be seen as a complement, not a substitute, to traditional association rule mining. There are three clear limitations: scalability; the limitation of a single hidden variable in the antecedent of each rule; and the adherence to a particular linear parametric family. However, it does provide extra information that typical association rule mining methods cannot replicate. As future work, we want to investigate how dynamic programming can be used to scale up the method, and how to better pre-process the data (e.g., by finding which marginally independent variables can be efficiently separated). Moreover, there are different sets of assumptions one can make in order to identify hidden variables [5]. To conclude, we hope that the ideas discussed in this paper can spark a new generation of algorithms for rule mining with hidden variables.

References

1. D. Bartholomew, F. Steele, I. Moustaki, and J. Galbraith. *The Analysis and Interpretation of Multivariate Data for Social Scientists*. Arnold Publishers, 2002.
2. C. Borgelt and R. Kruse. Induction of association rules: Apriori implementation. *15th Conference on Computational Statistics*, 2002.
3. W. Buntine and A. Jakulin. Applying discrete PCA in data analysis. *Proceedings of 20th Conference on Uncertainty in Artificial Intelligence*, 2004.
4. J. Gibson. *Freedom and Tolerance in the United States*. Chicago, IL: University of Chicago, National Opinion Research Center [producer], 1987. Ann Arbor, MI: Inter-university Consortium for Political and Social Research [distributor], 1991.
5. R. Silva. Automatic discovery of latent variable models. *PhD Thesis, Machine Learning Department, Carnegie Mellon University*, 2005.
6. R. Silva, R. Scheines, C. Glymour, and P. Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7:191–246, 2006.
7. C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery*, 2000.

A Data Mining Approach to the Joint Evaluation of Field and Manufacturing Data in Automotive Industry

Christian Manuel Strobel[1] and Tomas Hrycej[2]

[1] Universität Karlsruhe, Germany
mstrobel@statistik.uni-karlsruhe.de
[2] DaimlerChrysler AG, Research and Technology, 89081 Ulm, Germany
Tomas.Hrycej@daimlerchrysler.com

Abstract. The manufacturing quality can be evaluated only by considering the failure behavior of the product in the field. When relating manufacturing events to failure events, the main challenge is to master the huge number of combinations of both event types, of which each is only covered by a small number of occurrences. Additionally, this leads to the problem of selection of interesting findings - the appropriateness of the selection criterion for consequent decision making is a critical point. Another challenge is the necessity of mapping the process of manufacturing tests to a vector of variables characterizing the manufacturing process. The solution presented, focuses on correct rule generation and selection in the case of combinations with low coverage. Therefore statistical and decision theory approaches were used. The multiple hypothesis aspect of the rule set has also been considered. The application field was quality control of electronic units in automotive assembly, with thousands of variables observed.

1 Introduction

In the automotive assembly, testing of individual functions is frequently embedded in the assembly process. The focus is on testing the functionality of electronic units, whose number amounts, in the premium car segment, are up to several hundreds.

The ultimate criterion of manufacturing quality is the behavior of the product in the field. This is why it's important to learn as much as possible about the relationships between the assembly and testing procedure on the one hand, and the field quality measures on the other hand. In this case the task is to examine two different databases, each representing one part of the product life cycle: the assembly database, which stores the results of the electronic unit tests during the manufacturing process, and the field failure data collected in a company-wide service and warranty database.

The relationships between the assembly process and the field failure can be captured, in a the classical Data Mining manner, by association rules for which exist numerous algorithms. However, several fundamental characteristics of our application made some additional developments necessary:

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 625–632, 2006.
© Springer-Verlag Berlin Heidelberg 2006

1. There is a considerable number of assembly process features and failure types. Even in large event data bases, each event combination is very scarce, which leads to the necessity to evaluate the statistical properties of rules based on small samples. Also it's not necessary to consider complex combinations of three and more variables, because the support can be expected to tend to zero.
2. The interestingness of results found has to be viewed in economic terms: interesting rules are those, that have a large economic potential.
3. The large number of process features and failure types typically leads to a large number of rules found. Even with rule selection criteria considering the statistical significance, a certain amount of the rules found are random. Therefore, to have an idea about the reliability of the results, a rough assessment of the proportion of such random rules is desirable.

These aspects are specific, but not restricted to this application. Many large scaled industrial applications exhibit similar characteristics.

2 Process Model of Manufacturing Tests

The testing process for electronic units in the automotive industry is rather complex. Each car is tested several times during the assembly process to ensure an error-free delivery to the costumer. During each test, a pre-defined, assembly-status based scale of electronic units and their individual and interrelated functionality are tested. Each test is either performed automatically or with the additional input of a assembly worker and has basically two outcomes: either positive (without errors but eventually with warnings), possibly accompanied by the list of warnings, or negative, accompanied by the list of errors. Therefore, if a particular test results in a warning or error message of the tested devices, several consequent actions may be taken i.e. ignoring the message, repeating the test, taking the car out of the assembly line (so called rework) and so on.

This variety of procedures implies, that the same test may be performed more than once, with different results, each of which has to be interpreted individually. In order to combine and compare the field results with the test results, it is necessary to transform the test information from the manufacturing process into a binary representation, holding relevant information from the whole testing process of a specific electronic unit in a car during assembly - the so called *history of a electronic unit*. It captures the information which error was reported[1], whether this error emerge in other tests or not, did this error lead to rework and did this error appear as a warning in another test. Additionally, the production weekday and month are logged to capture time fluctuations of the quality.

3 Rule Generation

To improve the assembly process and its testing procedure, the relationship between the attributes of the assembly and testing process on the one hand, and the

[1] In the binary coding: has a given error appeared or not.

field failures on the other hand has to be clarified and hence can mathematically be described by association rules.

Let $I = \{i_1, i_2, ..., i_m\}$ and $J = \{j_1, j_2, ..., j_m\}$ be sets of items or literals. Let D be a transaction database over $I \times J$ where each transaction $T = (T_a, T_b)$ is a subset of $I \times J$. Further let A be a set of items with $A \subseteq T_a$. The implication of the from $A \rightarrow B$ with $A \subseteq I$ and $B =\subseteq J$ is called *association rule*. The *support*[2] of a rule $A \rightarrow B$ can be defined as the numbers of transactions containing A and B ($n_{A \wedge B}$) beeing $n_{A \wedge B} := |\{T \in D : A \cup B \subseteq T\}|$, $n_A := |\{T \in D : A \subseteq T_a\}|$, $n_B := |\{T \in D : B \subseteq T_b\}|$ and $n := |D|$.

However, the causes of a failure may be more complex: Boolean functions of multiple assembly terms describe the influence. Sophisticated algorithms exist for solving this *frequent item set problem* [2,5]. Considering the fact that even the occurrences of simple terms are scarce, the support of simple rules of type $A \rightarrow B$ can be expected to be very low. This makes the probability of finding complex rules of three and more antecedent terms with a significant support to vanish. Instead, all pairs of significant (see the next section) simple rule pairs $A_i \rightarrow B, A_j \rightarrow B$ have been combined to rules of the form $f(A_i, A_j) \rightarrow B$ with f being some of 8 possible nontrivial Boolean functions of two arguments, having higher significance[3] than each simple rule.

4 Rule Significance

In machine learning community, it is usual to select interesting rules with help of measures such as support, confidence, lift or combinations of both [8]. This approach may be adequate for a support exceeding some substantial count such as 30. However, many applications, including the ours, lack this property as obvious from the right graph in Fig. (2). For example in quality monitoring, early warning is essential. The number of failures occurred grows with the time interval of observation. The same rules might have been found earlier, if the observation interval would have been shorter, and it's thus always preferable, to use the shorter interval. Consequently, interesting rules cannot, per definition, exhibit a high support - it is necessary to discover them with a lower support possible. This is why statistical measures of significance have to be used for application of this type.

A widely used measure is the χ^2-measure of mutual dependence. Unfortunately, this measure can answer only the question how certain it is, that there is *some* dependence between the variables, no matter how strong this dependence is. What is really needed, is a statistically founded statement about the strength of dependence, which would allow us to identify strong and significant influences on the failure rates. One possibility are interval estimates of interestingness measures. An example for this is what we have called ,,safe lift" - an interestingness

[2] Mostly the support of a rule is defined as fraction of transactions (see i.e. [6,1]). In our application we always refer to the number count when talking about support.

[3] Here significance refers to an appropriated interestingness measure as explained in the following sections.

measure based on the well known lift. It can be expressed with help of sample counts (see Section (3)) and substituting sample probabilities for real ones:

$$L(B|A) = \frac{P(B|A)}{P(B)} = \frac{\frac{n_{A\wedge B}}{n_A}}{\frac{n_B}{n}} \tag{1}$$

Obviously, neither $P(B|A)$ nor $P(B)$ can be determined exactly, since both are computed from sample counts ($n_{A\wedge B}, n_A$, n_Bandn respectively). Assuming that n_B and n are sufficiently large (which is frequently satisfied), we can confine ourselves to an interval estimate of $P(B|A)$, given sample counts $n_{A\wedge B}$ and n_A.

The confidence interval estimate for the posteriori conditional probability based on Gaussian approximation of binomial distribution (with sample count n_A, number of positive outcomes ($n_{A\wedge B}$)) is the Wilson confidence interval $[T^l_{n_A}(conf(A \to B)), T^u_{n_A}(conf(A \to B))]$ (see [4]):

$$T^l_{n_A} = \frac{1}{1 + \frac{q^2_\alpha}{n_A}} \left(p + \frac{q^2_\alpha}{2n_A} - q_\alpha \sqrt{\frac{p(1-p)}{n_A} + \frac{q_\alpha}{4n^2_A}} \right) \tag{2}$$

with $q_\alpha := u_{1-\frac{\alpha}{2}}$ (the quantile of the standard normal distribution for given significance level α) and $p := \frac{n_{A\wedge B}}{n_A}$ being the empirical posterior conditional probability. For sufficiently large n_A, we can say $\frac{1}{1+\frac{q^2_\alpha}{n_A}} \to 1$, $\frac{q^2_\alpha}{2n_A} \to 0$, $\frac{q_\alpha}{4n^2_A} \to 0$ and therefore the lower interval bound (2) simplifies to $p - q_\alpha \sqrt{\frac{p(1-p)}{n_A}}$. The "safe lift" can thereby be defined as the ratio of the simplified lower interval bound and the posteriori probability of the consequence[4].

5 Decision Oriented Rule Selection

To select the *best* rules within the founded rules, the goal of the application has to be considered. One possibility is the attempt, to quantify the benefit from knowing the rule.

Each rule concerning a field failure gives a hint to it's causes and can be used to take an action ro reduce the failure rate. In an idealized setting, the benefit of such an action can be quantified i.e. in monetary terms. Usually, these effects can only be quantified partially. The effectiveness of the action taken cannot be predicted exactly, but a reasonable assumption may be, that the action may reduce the conditional failure rate to the level of the unconditional rate. For example, if the failure rate of reworked cars is higher than an average one, the reworking process may be scrutinized to reach at least the average performance. Then, the total savings through the failure rate reduction would amount to:

$$n_A \left(P(B|A) - P(B)\right) c_B = \left(n_{A\wedge B} - \frac{n_A n_B}{n} \right) c_B \tag{3}$$

with c_B being the unit costs of the failure B - or so called potential.

[4] In [7,10] a similar interesting measure was introduced as a kind of correlation factor.

The quantification of the action costs will scarcely be feasible in advance, since the actions themselves will be mostly designed only after viewing the rules. This is why the action costs will not be able to be included in the decision measure, and so we did in this application.

6 Multiple Hypothesis Problem

For an application as the present one, it is characteristic that the total number of significant findings can be very high. Therefore, a specific problem becomes particularly serious: the multiple hypothesis problem.

Rules received from a sample data set are a result of a random process. This randomness is quantified by significance measures. Accepting a rule on a certain significance level means that it is highly improbable that the law described by the rule does not exist. Nevertheless, the rule still *might* be a result of pure chance - only the probability of this is low. If our goal was to accept or reject all rules together, the procedure of testing individual rules against a given individual significance level was completely faulty. If each rule was a result of a rejection of a null hypothesis on a significance level π, the probability, that at least one rule is still a random product (while its null hypothesis is valid) is $\pi = 1 - (1 - \alpha)^n$. Vice versa, to ensure that the totality of rules are valid on the given level π, the significance level of individual rules would be $\alpha = 1 - (1 - \pi)^{\frac{1}{n}}$, a drastically low figure[5] for n of the order or magnitude of hundreds or thousands. Fortunately, we do not have to be so ambitious. A good rule set is such, that we can trust at least a significant portion of it. But it is still desirable to assess how large the portion of trustworthy rules really is - otherwise we would risk there are none.

The qualitative relationships within this problem can be illustrated with help of hypotheses about Gaussian distribution. Suppose the null hypothesis is, that a sample is drawn from the distribution $N(0, 1)$ while the alternative hypothesis postulates $N(m, 1)$. For a given significance level α, if the sample value is greater than the quantile q_α (such that $1 - F(q_\alpha) = \alpha$), the probability of this sample value if the sample is drawn from the null-hypothesis distribution $N(0, 1)$ is α while the corresponding probability if the sample is drawn from $N(m, 1)$ is $\gamma = 1 - N(q_\alpha - m, 1)$. The values of γ in dependence from α for some mean values of the alternative hypothesis m are given in left graph of Fig. (1) and the the ratio of γ/α in the right graph respectively. Obviously, the ratio grows with diminishing α. This is exactly what we expect: for higher significance level, the separation between correctly identified alternative-hypothesis samples, and wrongly identified null-hypothesis samples improves.

Going back to the case of discovering rules, suppose there are N rules whereof M are valid. Consequently $N - M$ rules are invalid, corresponding to the null hypothesis of no dependence between the antecedent and consequent. With a significance level of α, we can expect, in average, the testing algorithm to deliver $E_i = (N - M)\alpha$ invalid, and $E_v = M\gamma$ valid rules. In reality, neither M nor γ is known in advance. However, if the algorithm proposes K rules, we know that

[5] In literature this value is often referred to as Bonferroni correction [3,9].

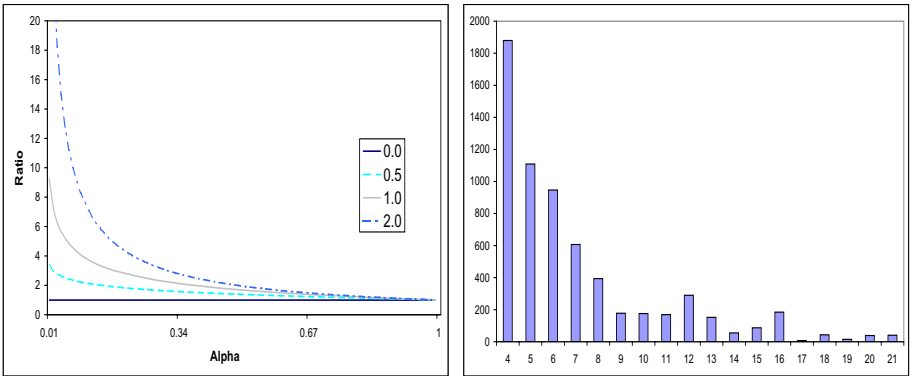


Fig. 1. Left: Ratio γ/α for different mean values of the alternative hypothesis m. Right: Histogram of the support for simple rules applying the number counts from Table (1).

this corresponds to the sum $K = (N - M)\alpha + M\gamma = N\alpha + M(\gamma - \alpha)$. Since $N\alpha > (N - M)\alpha$, we can expect a minimum of $K - N\alpha$ rules to be valid. Thus the ratio $\frac{K}{N\alpha}$ gives a good idea of proportion of usable rules in the rule set.

More precisely, we have estimated the number of invalid, randomly generated rules with help of its expected value $(N-M)\alpha$. $N\alpha$ was the upper bound of this estimate. For large values of $(N - M)\alpha$, the sample fluctuations around the expected value are small. For small $(N-M)\alpha$, an interval estimate is more reliable. The number of invalid rules generated by the algorithm is binomial distributed with sample size $N - M$ and probability α. The standard deviation of this number is $\sqrt{(N - M)\alpha(1 - \alpha)} \approx \sqrt{(N - M)\alpha}$. So the upper interval boundary is approximately $(N - M)\alpha + q\sqrt{(N - M)\alpha}$, with q being some quantile. Since we are ignorant of the value of M, the upper bound to be compared with the total number of rules found K is $L(\alpha) := N\alpha + q\sqrt{N\alpha}$.

7 Computing Results and Conclusion

There was a considerable number of variables in the described automotive test application with 3789 field failure (goal variables) and 3310 process attributes (influence variables). The aim of our application was to find relevant rules within this domain that are useful for the quality improvement process. Therefore there were 15,520,749 simple rules to examine and following the description of the previous section and applying the constraints from Table (1) we obtained a set of 7441 rules of which 6910 were simple and 525 complex rules.

The low support of complex rules is striking (see left graph of Fig. (2)), which seems to substantiate the commitment to a maximum of two antecedent variables (three or more influence variables would hardly produce a non-trivial support). Also the distribution of lift based on the contingence table (see right graph of Fig. (2)) shows, that there is a considerable number of seemingly valid rules,

Table 1. Rule pre-selection constraints used in the field feedback application

Constrains	Value	Constrains	Value	Constrains	Value
$min_{n_{A \wedge B}}$	4	Factor for std. deviation	1	α for safe lift	0.2
$min_{n_A} = min_{n_B}$	4	Factor for composed rules	1.5	minimum safe lift	1.5

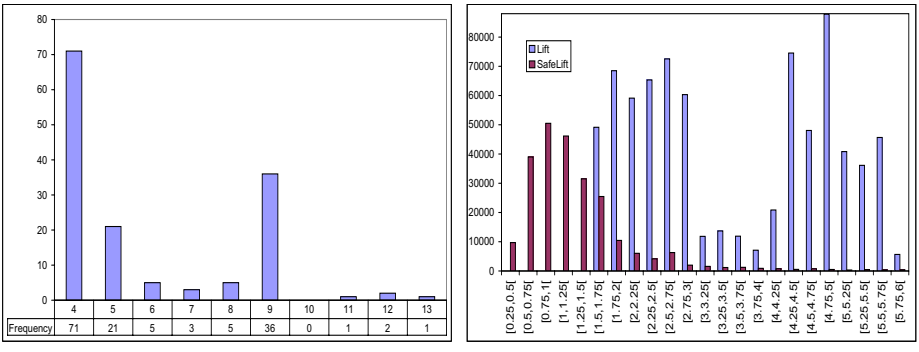


Fig. 2. Left: Histogram for $n_{A \ B}$ for significant complex rules Right: Histogram of lifts and safe lifts for significant rules

having a lift ≥ 1.5. However, the histogram of safe lift (Fig. (2)) exhibits, that the bulk of these rules are hardly statistically significant as only 80% of these rules have a safe lift ≥ 1.0. This shows, how important it is, to consider the statistical properties and justifies the pre-selection criterion of rules based on the safe lift.

The search for economically useful rules makes a trade-off between the extensive (considering many rules) and intensive (only the most important rules) approach necessary. This can be reached by the potential measure defined in Section 5. Ordering and accumulating the rules by their potential, results into a marginal utility function, which allows to set an economically meaningful cut-off point. The slope of this function can be compared with the estimated average costs of considering the rule and taking a corrective action. The cut-off point would have to be set at the potential value for which the slope and the costs are equal.

The multiple hypothesis problem can be illustrated by a selected subset of rules for a particular electronic unit. From the total number of 218363 possible rules of this subset, only 2900 were considered as candidates[6]. Using the constraints listed in Table (1) we found 61 rules. These rules were significant on different levels. For a given significance level, we can compute the expected mean number, and the upper bound of ,,random" rules, as discussed in the previous section. For example, there were 35 rules with $\alpha > 0.005$. From these rules, at

[6] Applying $n_A > 10$ and $n_B > 4$ reduced the overall amount of rules to the candidate rules.

most 17 can be expected to be random, using the last formula of Section 6. This is a considerable number, but still an acceptable proportion.

Concluding, in this work we have presented a framework for the analysis of quality data, with the goal to find relationships between the assembly and testing process and the failures in the field. The basic method was the search for association rules. However, several characteristics of our application lead to extensions of mainstream methods:

1. It was necessary to map the assembly process data to a feature vector, with help of expert knowledge.
2. The considerable number of variables involved lead to high number of value combinations and thus to low supports of individual candidate rules. This made considering the statistical properties of rule selection criteria necessary, to avoid the risk of obtaining random rules.
3. Due to the industrial context, the rules selection criteria had to consider the economic impact of the rules.
4. The large number of rules obtained by the sample, justifies the question how many of them are really valid and which are random. We used a simple framework to determine these proportions and showed that a large part of the rules founded are valid.

Since these characteristics are shared by many industrial applications, the solutions presented may be useful for a broad scope of Data Mining problems.

References

1. C. C. Aggrawal and P. Yu. Online generation of association rules. *ICDE*, 1998.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proceedings 20th Int. Conference on Very Large Data Bases*, pages 487–499, 1994.
3. Y. Benjamini and H. Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Statist. Soc.*, B:289–300, 1995.
4. T. T. D.-A. Brown, L. D. Cai. *Statistical Science, 16(2)*, pages 101–133, 2001.
5. B. Goethals. Survey on frequent pattern mining. HIIT Basic Research Unit Department of Computer Science University of Helsinki, Finnland.
6. M. K. Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.
7. G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. *Knowledge Discovery in Databases*, pages 229–248, 1991.
8. J. D. U. S. Brin, R. Motwani and S. Tsur. Dynamic itemset counting and implication rules for market basket data. *AIn Proc. of the ACM SIGMOD Intl Conf. on Management of Data (ACM SIGMOD 1997)*, pages 265–276, 1998.
9. B. Walsh. Multiple comparisons: Bonferroni corrections and false discovery rates. Lecture Notes for EEB 581, May 2004.
10. C.-Y. W. Wei-Chou Chen, Shian-Shyong Tseng. A novel manufacturing defect detection method using data mining approach. *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 2004.

Incremental Aspect Models for Mining Document Streams

Arun C. Surendran[1] and Suvrit Sra[2]

[1] Microsoft Research, 1 Microsoft Way, Redmond, WA 98052, USA
acsuren@microsoft.com
[2] Dept. of Computer Sciences, The University of Texas at Austin, TX 78712, USA
suvrit@cs.utexas.edu

Abstract. In this paper we introduce a novel approach for incrementally building aspect models, and use it to dynamically discover underlying themes from document streams. Using the new approach we present an application which we call "query-line tracking" i.e., we automatically discover and summarize different themes or stories that appear over time, and that relate to a particular query. We present evaluation on news corpora to demonstrate the strength of our method for both query-line tracking, online indexing and clustering.

1 Introduction

In this paper we present a new unsupervised framework for mining a stream of documents to extract and summarize the set of underlying themes. The system discovers and isolates themes one by one using a novel approach that combines EM and functional gradient methods. We show that our algorithm naturally leads to incorporating a HITS-like spectral technique within a probabilistic framework, thereby combining the power of both to create a unique document mining framework. The most important functionality of our approach is its ability to handle streaming data without having to retrain the entire model. As a result, the model can grow or shrink as needed leading to a faster, scalable system that permits easier model selection as compared to a batch system.

Based on our incremental framework we present a new application called "queryline tracking" which collects all documents relating to a query over time, and automatically groups and summarizes these into themes. The system automatically keeps track of themes that a user has seen and alerts the user to new themes not seen by him/her, as soon as they are discovered. We further show that our system makes meaningful summaries that correlate well with human concepts, and also has good indexing properties (the model preserves the original distances between documents as much as possible).

2 Incrementally Built Aspect Models (BAM)

BAM is motivated by ideas from density boosting [15,16], incremental EM [15], and aspect models [5]. In spite of the word "boosting" in the title, density

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 633–640, 2006.
© Springer-Verlag Berlin Heidelberg 2006

boosting (also referred to as unsupervised boosting) is closer to semi-parametric maximum likelihood methods [3] than to traditional boosting.

Let the input be a set of documents $D = \{d_1, d_2, \ldots, d_N\}$, where each document is represented by an M-dimensional vector of words taken from a vocabulary $W = \{w_1, w_2, \ldots, w_M\}$. The data is represented by a word-document co-occurrence matrix of size $M \times N$. The frequency of word w in document d is given by n_{wd} (or appropriately weighted versions of it like *tfidf*). Both M and N can vary as new documents are added or older documents are deleted.

The aspect model is a latent variable model [5] where each document is a mixture of underlying aspects, or themes. Each theme is represented by an the distribution of words $p(w|z)$. In PLSI [5] the word and document probabilities are conditionally independent given the aspect z. The joint word-document probability (with K hidden aspects) is

$$F_K(w,d) = \sum_{k=1}^{K} P(z_k)P(d|z_k)P(w|z_k) = (1-\alpha)F_{K\ \ 1} + \alpha h_K, \tag{2.1}$$

where $\alpha = P(z_k)$ gives the prior probability that any word-document pair belongs to the K_{th} aspect z_K. Thus, given the current model $F_{K-1}(w,d)$ we wish to compute h_K (the distribution for the individual aspect K) and α (the combining parameter), to obtain F_K using (2.1). Sometimes we will abuse the notation and also refer to h_K as the aspect or latent variable.

A natural objective function for this estimation is the empirical log-likelihood $L_K = \sum_{wd} n_{wd} \log F_K(w,d)$, which may be written as,

$$L_K = \sum_{w,d} n_{wd} \log\big((1-\alpha)F_{K\ \ 1}(w,d) + \alpha h_K(w,d)\big) \tag{2.2}$$

$$\geq \sum_{w,d} n_{wd} \left[(1-p_{wd}) \log \frac{(1-\alpha)F_{K\ \ 1}}{(1-p_{wd})} + p_{wd} \log \frac{\alpha h_K}{p_{wd}}\right] \equiv Q(\tilde{P}, h_K, \alpha), \tag{2.3}$$

where $\tilde{P} = \{p_{wd}\}\ \forall\ w,d$, and (2.3) provides a "surrogate function" that lower-bounds L_K, and can be maximized instead. Thus, the E-step maximizing (2.3) over p_{wd} is

$$p_{wd} = \frac{\alpha h_K(w,d)}{(1-\alpha)F_{K\ \ 1}(w,d) + \alpha h_K(w,d)}. \tag{2.4}$$

Using (2.4) in (2.3) we get $Q(\tilde{P}, h_K, \alpha) = \sum_{w,d} n_{wd} E_{\tilde{P}}\big[\log F_K(w,d)\big]$. In the traditional M-step h_K is estimated such that this Q-function is maximized. Alternatively, we can perform a first-order functional gradient ascent on Q. This idea is similar to the one used in AnyBoost [11], density boosting [16] and semi-parametric methods [3]. To that end, we approximate the difference $L_K - L_{K-1} \approx \sum_{w,d} n_{wd}\alpha\langle \nabla L, h_K - F_{K-1}\rangle$, where $\nabla L = \frac{\partial L(F_{K-1} + \delta \mathbf{1}_{wd})}{\partial \delta}\,|_{\delta=0}$ is the functional derivative of L, and $\langle \nabla L, h_K - F_{K-1}\rangle$ is the directional derivative in the new direction $h_K - F_{K-1}$. To ensure an increase in Q, it is enough to maximize the expected value of the difference $L_K - L_{K-1}$. This leads to a generalized EM approach, wherein the objective function is increased using a unique

combination of EM and functional gradient based approaches. Consequently, we call this new method *Expectation Functional Gradient (EFG)*.

If a data point (w, d) is well explained by the existing model, then, $F_K = F_{K-1}$, and the corresponding directional derivative is zero. Thus $\langle \nabla L, h_K - F_{K-1} \rangle$ is non-zero only for points well represented by h_K. For the log-likelihood in (2.2) the expected change in Q, $E_{\tilde{P}}[L_K - L_{K-1}]$ equals

$$\alpha \sum_{w,d} n_{wd} p_{wd} \langle \nabla L, h_K - F_{K\ 1} \rangle = \alpha \sum_{w,d} n_{wd}\ p_{wd} \left(\frac{h_K(w,d) - F_{K\ 1}(w,d)}{F_{K\ 1}(w,d)} \right). \quad (2.5)$$

We can now maximize $E_{\tilde{P}}[L_K - L_{K-1}]$ by solving

$$h_K = \operatorname*{argmax}_h \sum_{w,d} n_{wd}\ p_{wd} h(w,d) / F_{K\ 1}(w,d). \quad (2.6)$$

Once h_K has been estimated, α can be estimated using line search on (2.2). We note that as a natural consequence of the above maximization steps, the quantities $1/F_{K-1}(w,d)$ act like weights—data points that are **well represented** by the current model tend to be **down weighted**, and data points that are **poorly represented** tend to be **given more attention** in the next BAM step. This procedure is similar to the one used in boosting.

Estimating h_K: Recall that $h_K(w,d) = p(w|z)p(d|z)$. Let $\boldsymbol{w} = p(w|z_K)$ over all words w, and $\boldsymbol{d} = p(d|z_K)$ over all documents d. Introducing the matrix $\boldsymbol{V} = [n_{wd}\ p_{wd}/F_{wd}]$ we write (2.6) as

$$\min_{\boldsymbol{w},\boldsymbol{d}} \quad Q(\boldsymbol{w},\boldsymbol{d}) = -\boldsymbol{w}^T \boldsymbol{V} \boldsymbol{d}, \qquad \text{where} \quad \boldsymbol{w},\boldsymbol{d} \geq 0. \quad (2.7)$$

However, without further constraints on $\boldsymbol{w}$ and $\boldsymbol{d}$ (2.7) is unbounded. Using a constraint on the L_1 norm of the vector (which gives it a probabilistic interpretation) leads to a formulation similar to PLSI (discussed later). Alternatively we could use a regularization restricting the magnitude of the $\boldsymbol{w}$ and $\boldsymbol{d}$ vectors, and then later re-interpret them as probabilities. This can be done in a principled way starting with (2.2) to incorporate a regularizer. The rest of the analysis will remain unchanged, but the regularized form of the function to be minimized in (2.7) can be modified to be [1]

$$Q(\boldsymbol{w},\boldsymbol{d}) = -\boldsymbol{w}^T V \boldsymbol{d} + \nu(\boldsymbol{w}^T \boldsymbol{w}) + \mu(\boldsymbol{d}^T \boldsymbol{d}).$$

where μ and ν are regularization parameters. Differentiating $Q(\boldsymbol{w},\boldsymbol{d})$ with respect to $\boldsymbol{w}$ and $\boldsymbol{d}$ and setting the derivatives to equal zero we obtain the system of equations

$$\boldsymbol{w} = \frac{\boldsymbol{V}\boldsymbol{d}}{\nu}, \qquad \boldsymbol{d} = \frac{\boldsymbol{V}^T \boldsymbol{w}}{\mu}, \quad (2.8)$$

which can be solved iteratively. Setting $\boldsymbol{w}$ and $\boldsymbol{d}$ to the left and right singular vectors (with $\nu,\ \mu = 1$) of $\boldsymbol{V}$ provides the solution. Readers will notice the similarity of these steps to the popular HITS algorithm [7] which is a spectral

[1] If $\boldsymbol{w},\boldsymbol{d}$ are initialized to be positive, they will stay positive, and the solution will satisfy the non-negativity constraints.

method. Due to its similarity to a spectral algorithm we call (2.8) the *spectral M-step*. Since both $\boldsymbol{w}$ and $\boldsymbol{d}$ are non-negative, we can normalize them and interpret them as probabilities. We obtain $h_K = \boldsymbol{w}\boldsymbol{d}^T$ as the new aspect. Thereafter, we determine α using line search and then obtain the updated model $F_K = (1-\alpha)F_{K-1} + \alpha h_K$.

2.1 Understanding How BAM Works

Relation to PLSI: In PLSI, the traditional M-step leads to the solution $p(w|z) \propto \sum_d n_{wd}p_{wd}$ and $p(d|z) \propto \sum_w n_{wd}p_{wd}$, which can be re-written as

$$\boldsymbol{w} \propto \boldsymbol{V}\mathbf{1}, \qquad \boldsymbol{d} \propto \boldsymbol{V}^T\mathbf{1}, \tag{2.9}$$

where $\boldsymbol{V} = [n_{wd}p_{wd}]$. Comparing this with (2.8) we can see that this M-step (2.9) is essentially one spectral step with $\boldsymbol{w}$ and $\boldsymbol{d}$ initialized to $\mathbf{1}$ (and an unweighted $\boldsymbol{V}$). The regular PLSI M-step simply computes an average of the pre-weighted data. Due to this behavior, in an incremental setting, PLSI leads to a sequence of averaged models, whereas BAM produces a sequence of coherent topics. It is difficult to grow PLSI models to accommodate new topics, while BAM is built to do so. One has to rerun PLSI with an increased number of aspects - increasing the computational needs with no guarantee that the new themes will correspond to the old ones, or that the old ones will be rediscovered. Model selection can also be a problem with PLSI—multiple runs with different model sizes are needed. Computationally BAM is faster since, unlike PLSI, it does not need annealing to work well, plus spectral algorithms tend to exhibit fast convergence properties.

Following (2.8), we suggest a way to modify PLSI to grow the model - before estimating a new aspect, the data is weighted by $1/F_{K-1}$. Then the regular M-step is replaced by the spectral M-step, i.e., iterate as per (2.8) until convergence. This is equivalent to replacing $\boldsymbol{V}$ in (2.8) by $\boldsymbol{V} = [n_{wd}p_{wd}]$. In practice, both these approaches seem to yield similar results, so in this paper we use the latter.

Relation to HITS: The updates in (2.8) are strikingly similar to another very popular algorithm used in the IR community to rank a set of web documents into authorities and hubs—HITS [7]. BAM uses a similar idea to rank words and documents. HITS can be shown to discover tightly knit clusters (TKC) based on link structure [9]; similarly, BAM discovers TKCs based on how strongly words and documents are connected to each other, which is what we want to achieve. Thus BAM looks at the data weighted by $1/F$ (which defines the search space), and then finds an appropriate cluster within this space. We can think of these two steps as *restriction* and *discovery* steps respectively. This process is repeated till all relevant topics are found. BAM also reduces the chances of mixing up weakly connected components, i.e., it reduces the chances of discovering mixed topics. The TKC issue can be a problem for HITS because sometimes the best cluster is not the most relevant one. For BAM, this is not a problem; in fact it is an advantage. BAM can be thought of as performing a series of soft cuts on the bipartite graph to extract many tightly-knit overlapping components from it.

Relation to other spectral graph partitioning approaches: There are other ways to partition these graphs, e.g. normalized cut using the Fiedler vec-

tor (eigenvector corresponding to the second smallest generalized eigenvalue [4]. BAM uses the top left and right singular vectors. Norm-cut looks at both inter- and intra-partition properties while our algorithm is thought to target the tightness of the partition in question.

The convergence properties of BAM depend on the nature of the weighted word-document graph at each BAM step. Due to lack of space, we defer a detailed analysis of stability and convergence to future publications.

Other Relevant Work: We have briefly discussed BAM's relation to PLSI. Latent Dirichlet allocation (LDA) [2] addresses many of the issues faced by PLSI, including the ability to grow with data [2,6], but at commensurate additional computational costs. BAM could be extended to LDA, something that we defer to the future. LSI is a spectral method to finding topics but lacks generative semantics. There exist some incremental approaches to LSI (e.g. [1])

There has been a significant amount of work in topic tracking and detection [8] but we only mention a couple that are very closely related to our method. The work by Kumar, 2004 [14] uses non-probabilistic, graph theoretic ideas to extract storylines. It cannot handle overlapping categories, and is not dynamic or incremental. Another recent work analyzes chunks of data over time using static PLSI-like models, but tracking is done using similarity of aspects across time scales [12].

2.2 Handling Streaming Data

To handle streaming data, first we need to understand how much of the new data is already explained by the existing models. We use a "fold-in" approach similar to the one suggested in [5]. For each aspect z_k we keep $p(w|z_k)$ values fixed for all the words that are already seen. We then use the spectral step to estimate the probabilities of the new words (the $p(w|z)$ vectors are normalized as needed), and the document probabilities $p(d|z_k)$. Using the estimated probabilities we compute a new F for all the new data, and use this new F as a starting point to discover new themes as needed.

3 Experiments and Results

We present some preliminary results on news corpora to demonstrate the performance of our algorithm in queryline tracking, indexing and clustering.

3.1 Tracking Storylines Around a Query

We present an interesting application of BAM, called *Queryline Tracking*, which is a mixture of TREC tasks like filtering and novelty detection, but is centered around a query. Essentially, this task involves discovering and tracking themes or storylines [14] based upon a query. As new data comes in, we only wish to surface new themes. We demonstrate this idea on the publicly available RCV1-v2 Reuters news corpus (23,000 documents) [10]. We use data from the first 10 days (Aug 20-30 1996) and run BAM on it incrementally, one day at a time.

Table 1. BAM captures all the storylines around the news alert "Clinton" over a period of 10 days. Days 4 and 6 did not have enough articles

Day 1 Storylines					Day 2	Day 3
Dole	Wage	McDougal	drug	Gingrich	McCurry	Zogby
Powell	Clinton	Whitewater	Hatch	terror	Yeltsin	vote
Clinton	Minimum	Susan	Marijuana	nuclear	Chechnya	gap
convention	bill	Arkansas	coppl	Libya	strenuous	poll
Kemp	legist.	sentence	McCurry	Iraq	Russian	narrow
Day 5	Day 7	Day 8	Day 9	Day 10		
FDA	train	read	Jackson	Morris	litigation	
tobacco	handgun	Jeep	Cuomo	resign	tobacco	
smoke	Brady	liter	Jesse	Dick	lung	
cigarette	Ohio	Americorp	welfare	prostitute	cancer	
advert	Huntington	Cherokee	disagree	tabloid	letter	

We demonstrate the idea using the query "Clinton". Each day we collect the documents that contain the word "Clinton". At the end of each day, we if we have less than 20 documents, we defer the documents to the next day. Otherwise we run BAM on the data to discover new storylines. We can see the results in Table 1.

The first day has stories about the presidential election, Clinton signing a bill to raise minimum wage, The Whitewater case, Senator Hatch complaining to the President about increase in drug use, and Gingrich cautioning the President that the country needs to be preemptively deal with external nuclear threats. On subsequent days the system discovers stories about Chechnya, Zogby's election tracking poll, Clinton asking the FDA to move against illegal practices in tobacco advertisement, etc. Items belonging to the older themes are subsumed by the older models and are available, but not surfaced. We can demonstrate the advantage of the incremental algorithm by showing that for the same number of topics discovered, the static model rediscovers themes from previous days.

We can show similar results using other queries (omitted due to lack of space). The rest of the experiments in the paper are on query independent tasks.

3.2 Indexing Power

Just like LSI, BAM can also be used as an indexing algorithm. We evaluate this by measuring how well it preserves distances between documents in the lower dimensions. We demonstrate this using 1-nearest neighbor (1-NN) comparison on the well known Reuters-21578 set. This dataset has 9063 documents in the training set and 3699 documents in the testing set, with 22226 words and spans 113 different topics. First, we build a K-aspect BAM model using the training data and then take the dot product of the data vector with each $p(w|z)$ vector. Hence the documents are projected from 22226 dimensional space to K dimensions. Now for each projected document d, we choose the nearest neighbor n and compare the labels of n against the true class labels for d. Let A be the number

Table 2. F_1 scores for the 113 class Reuters-21758 dataset using 1-NN. LMDS scores are from [13].

No. of	BAM		LMDS
dimensions	Testing data	Training	Training
10	0.584	0.606	0.625
20	0.677	0.697	0.714
50	0.724	0.757	0.754
100	0.750	0.777	0.768

Table 3. NMI and Rand Index scores for RCV12 and Reuters datasets, for BAM and PLSI. (Higher scores are better) signifies that these difference of these scores from those of BAM are statistically significant at the level of 0.02.

	RCV1-v2 subset		Reuters subset	
	NMI	Rand	NMI	Rand
BAM	0.54	0.49	0.56	0.32
PLSI	0.54	0.49	0.51	0.26

of matching labels over all documents, and B be the number of unmatched labels. The microaveraged F_1 score is then $F_1 = A/(A + B/2)$. For test data, we can follow a similar procedure except we choose the nearest neighbor from the *training set* in the reduced space. Table 2 shows the average F_1 numbers as K is varied from 10 to 100.

The F_1 score using the unprojected training data was 0.719 [13]. BAM matches this score when using around 50 aspects. The results on the unseen test data is similar to that on the training data. Training data results for BAM are comparable to Landmark MDS (which is similar to LSI) [13].BAM sacrifices a little bit of indexing power to gain the ability to grow, and to create higher quality topics.

3.3 Correlation of Aspects with Human Labels

The goal of this section is to quantitatively show that the aspects created by BAM correlate well with human labeling. These experiments are done on two selected subsets with 10 topics each - one from Reuters-21578 (8009 documents) and another from RCV1-v2 (14814 documents). We run BAM incrementally to get 10 aspects. Then, we do a hard classification for each document by assigning it to the aspect with highest $p(z|d)$ value. We then compute normalized mutual information (NMI) and the Rand index for the resulting partitions by making use of the topic labels assigned by human experts. We average over 10 runs to get the numbers shown in on Table 3. BAM performs significantly better than PLSI on the Reuters subset and performs just as well on the RCV1-v2 subset. The number of documents per class is similar across classes in the RCV1-v2 subset, and most clustering algorithms tend to do well on such sets. The Reuters subset is very unbalanced, and we see that BAM does better on this set.

4 Summary

In this paper we have introduced a new framework for building incremental aspect models incorporating the strengths of both spectral and probabilistic methods. The main advantage of this method is that it can handle documents arriving in a stream, and the model can grow or shrink as needed. We demonstrated some of the capabilities of the new approach in indexing and clustering. Using the new framework we presented a new application called "queryline tracking".

Acknowledgments. We thank John Platt, Chris Meek and Asela Gunawardana for valuable discussions.

References

1. R. K. Ando and L. Lee. Iterative Residual Rescaling: An Analysis and Generalization of LSI. In *SIGIR*, September 2001.
2. D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical Topic Models and the Nested Chinese Restaurant Process. In *NIPS*, 2004.
3. D. Böhning. A review of reliable maximum likelihood algorithms for semiparametric mixture models. *J. of Stat. Planning and Inference*, 47:5–28, 1995.
4. I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.
5. T. Hofmann. Unsupervised learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, 42, 2001.
6. Z. G. J. Zhang and Y. Yang. A Probabilistic Model for On-line Document Clustering with Application to Novelty Detection. In *NIPS*, 2005.
7. J. Kleinberg. Authoritative sources in a hyperlinked environment. *ACM*, 46(5):604–632, 1999.
8. A. Kontostathis et al. A survey of emerging trends detection in textual data mining. In *Survey of Text Mining*, pages 185–224, 2003.
9. R. Lempel and S. Moran. The stochastic approach for link-structure analysis (salsa). In *ACM Tran. on Info. Sys.*, volume 19, pages 131–160, 2001.
10. D. D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *JMLR*, 5:361–397, 2004.
11. L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting Algorithms as Gradient Descent in Function Space. In *NIPS*, pages 512–518, 2000.
12. Q. Mei and C. Zhai. Discovering evolutionary theme patters from text - an exploration of temporal mining. In *Knowledge Discovery and Data Mining*, pages 198–207, 2005.
13. J. Platt. FastMap, MetricMap, and Landmark MDS are all Nystrom Algorithms. In *10th International Workshop on AI and Statistics*, pages 261–268, 2005.
14. U. M. R. Kumar and D. Sivakumar. A graph-theoretic approach to extracting storylines from search results. In *Knowledge Discovery and Data Mining*, pages 216–225, 2004.
15. G. Ridgeway. Looking for lumps: boosting and bagging for density estimation. *Computational Statistics and Data Analysis*, 38(4):379–392, 1999.
16. S. Rosset and E. Segal. Boosting density estimation. In *NIPS*, 2002.

Learning Approximate MRFs from Large Transaction Data*

Chao Wang and Srinivasan Parthasarathy

Department of Computer Science and Engineering, The Ohio State University
srini@cse.ohio-state.edu

Abstract. In this paper we consider the problem of learning approximate Markov Random Fields (MRFs) from large transaction data. We rely on frequent itemsets to learn MRFs on the data. Since learning exact large MRFs is generally intractable, we resort to learning approximate MRFs. Our proposed modeling approach first employs graph partitioning to cluster variables into balanced disjoint partitions, and then augments important interactions across partitions to capture interdependencies across them. A novel treewidth based augmentation scheme is proposed to boost performance. We learn an exact local MRF for each partition and then combine all the local MRFs together to derive a global model of the data. A greedy approximate inference scheme is developed on this global model. We demonstrate the use of the learned MRFs on the selectivity estimation problem. Empirical evaluation on real datasets demonstrates the advantage of our approach over extant solutions.

1 Introduction

In this paper we address the problem of learning approximate *Markov Random Fields* (MRF) from large transaction data. Examples of such data include market basket data, web log data, etc. Such data can be represented by a high-dimensional data matrix, with each row corresponding to a particular market basket (web session), and each column corresponding to a particular item (web page). Each entry takes a value of "1" if the corresponding item is in the corresponding basket, otherwise it takes a value of "0". The data matrix is binary, and very often in such applications, highly sparse in that the number of non-zero entries is small.

To model such data effectively in order to answer queries about the data efficiently, we consider the use of probabilistic models. Probabilistic models capture association or causal correlations among attributes in data and have been successfully applied in applications such as selectivity estimation in query optimization [1,2,3], link analysis/recommender systems [4,5] and bioinformatics [6].

Specifically to tackle the selectivity estimation problem, Pavlov *et al.* [3] propose a *Maximum Entropy* (ME) model based on frequent itemsets. The ME model is essentially equivalent to an MRF and is effective in estimating query selectivity. However, a key limitation of their approach is that it needs to learn a *local model* over query variables

* This work is supported in part by the following research grants: DOE Award No. DE-FG02-04ER25611; NSF CAREER Grant IIS-0347662.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 641–649, 2006.
© Springer-Verlag Berlin Heidelberg 2006

on the fly for every query. Due to the fact that inferring an ME model is an expensive iterative process, such a just-in-time model construction approach is not appropriate in settings where online estimation time is crucial. The alternative is to first learn a *global model* offline. Subsequently the queries can be answered on the fly using standard probabilistic inference methods [7,8,9]. The advantages are a more accurate model (relies on complete information from all the data) and online performance. The critical challenge is that a global model may be prohibitive to compute for large datasets of high dimension (offline learning cost) . To address this problem, in this paper, we consider the problem of employing frequent itemsets to learn approximate global MRFs on large transaction data. Frequent itemsets capture important local distribution information of the data. Hollmen *et al.* [10] proposed to use frequent itemsets to learn mixture models on the local scale. Goldenberg *et al.* [5] proposed an approach (SNBS) of using frequent itemsets to learn large Bayesian networks.

The main contributions of this paper are summarized below: **a.** We introduce a novel divide-and-conquer style approach based on graph partitioning to learning approximate MRFs from large transaction data; **b.** We introduce a novel interaction importance and treewidth based augmentation scheme to capture interdependencies across partitions; **c.** We conduct an extensive empirical study on real datasets to show the efficiency and effectiveness of the new approach.

2 Background

Let $\mathcal{I}$ be a set of items, i_1, i_2, ..., i_d. A subset of $\mathcal{I}$ is called an *itemset*. The *size* of an itemset is the number of items it contains. An itemset of size k is a k-itemset. A transaction dataset is a collection of itemsets, $D = \{t_1, t_2, \ldots, t_n\}$, where $t_i \subseteq \mathcal{I}$. For any itemset α, we write the transactions that contain α as $D_\alpha = \{t_i | \alpha \subseteq t_i \; and \; t_i \in D\}$. In the probabilistic model context, each item is modeled as a random variable[1].

Definition 1. *(Frequent itemset). For a transaction dataset D, an itemset α is frequent if $|D_\alpha| \geq \sigma$, where $|D_\alpha|$ is called the support of α in D, and σ is a user-specified non-negative threshold.*

Definition 2. *(Markov Random Field). An Markov Random Field (MRF) is an undirected graphical model in which vertices represent variables and edges represent correlations between variables. The joint distribution associated with an undirected graphical model can be factorized as follows: $p(X) = \frac{1}{Z(\psi)} \prod_{C_i \in \mathcal{C}} \psi_{C_i}(X_{C_i})$, where $\mathcal{C}$ is the set of maximal cliques associated with the undirected graph; ψ_{Ci} is a potential function over the variables of clique C_i and $\frac{1}{Z(\psi)}$ is a normalization term.*

Using Frequent Itemsets to Learn an MRF. The idea of using frequent itemsets to learn an MRF was first proposed by Pavlov *et al.* [3]. A k-itemset and its support represents a k-way statistic and can be viewed as a constraint on the true underlying distribution that generates the data. Given a set of itemset constraints, a *Maximum Entropy* (ME) distribution satisfying all these constraints is selected as the estimate for the true

[1] In this article we use these terms – item, (random) variable – interchangeably.

```
Iterative-Scaling(C)
Input : C, collection of itemsets;
Output : MRF M;
1. Obtain all involved variables v and
   initialize parameters of M;
   //typically uniform over v;
2. while (Not all constraints are satisfied)
3.    for (each constraint C_i)
4.       Update M to force it to satisfy C_i;
5. return M;
```

Fig. 1. Iterative scaling algorithm

underlying distribution. This ME distribution is essentially equivalent to an MRF. A simple iterative scaling algorithm can be used to learn an MRF from a set of itemsets. Figure 1 presents a high-level outline of a computationally efficient version of the algorithm given by Jelinek [11]. Efficient inference is crucial to the running time of the learning algorithm. We call models learned through exact inference procedures *exact*.

The *junction tree* algorithm is a commonly-used exact inference engine for probabilistic models. The time complexity of the junction tree algorithm is exponential in the treewidth of the underlying model. For real world models, it's quite common that the treewidth will be well above 20, making learning exact models intractable. As a result, we have to resort to learning approximate models. One possible approach is to plug in approximate inference engines during the model learning process. However, it is not clear whether or not the learning process will still converge when subjected to approximate inference engines. In this paper, we pursue another approach – to learn a simplified model which is feasible to learn exactly and is close to the true exact model.

3 Learning Approximate MRFs

Before discussing our proposed approach, let us consider an extreme case in which the overall graph consists of a set of disjoint non-correlated components. Then the joint distribution can be obtained in a straightforward fashion according to Lemma 1.

Lemma 1. [2] *Given an undirected graph G subdivided into disjoint components D_1, D_2, ..., D_n (not necessarily connected components), and there is no edge across any two components, then the probability distribution associated with G is given by:* $p(X) = \prod_{i=1}^{n} p(X_{Di})$

3.1 Clustering Variables Based on Graph Partitioning

The basic idea of our proposed divide-and-conquer style approach comes directly from the above observation. Specifically, the variables are clustered into groups according to their correlation strengths. We call the group *variable-cluster*. Then a local MRF is defined on each *variable-cluster*. In the end we aggregate the local models to obtain a global model. From Lemma 1, we see that if we have a perfect partitioning of an MRF

[2] This follows immediately from the *global Markov property* of the MRF.

in which there is no correlations across different partitions, the divide-and-conquer style approach gives the exact estimate of the full model. Even for an imperfect partitioning, if the correlations across partitions are not strong, we still expect a reasonable approximation of the full model. Correspondingly, the first problem we face is how to cluster the variables such that the correlations across partitions is minimized.

***k*-MinCut.** The k-MinCut problem is defined as follows [12]: Given a graph $G = (V, E)$ with $|V| = n$, partition V into k subsets, $V_1, V_2, \ldots, V_k$ such that $V_i \cap V_j = \emptyset$ for $i \neq j$, $|V_i| = \frac{n}{k}$, and $\cup_i V_i = V$, and the number of edges of E whose incident vertices belong to different subsets is minimized. Given a partitioning P, the number of edges whose incident vertices belong to different partitions is called the *edge-cut* of the partitioning. In the case of weighted graphs, we minimize the sum of weights of all edges across different partitions.

The k-MinCut can serve our purpose of clustering variables. Each graph partition corresponds to a *variable-cluster*. Intuitively, we want to maximize correlations among variables within *variable-clusters*, and minimize correlations among variables across *variable-clusters*. So we should make the weight of edges reflect the strength of correlations between variables. We have the collection of all frequent itemsets. In particular, itemsets of size 2 specify the connectedness structure of the graph, and their associated supports indicate the strength of pairwise correlations between variables. We can use their supports as the edge weights directly. However, we also have higher-order statistics available, i.e., the larger itemsets. We expect that taking into consideration the information of all itemsets will yield a better weighting scheme. To this end, we propose an accumulative weighting scheme as follows: for each itemset, we add its support to all related edges, whose two vertices are contained by the itemset. Intuitively, we strengthen the graph regions which involve closely related itemsets in the hope that the edges within these regions will not be broken in the partitioning. An advantage of the k-MinCut partitioning scheme is that the resulting clustering is forced to be balanced. This is desirable for the sake of efficient model learning, since we will not encounter very large *variable-clusters* which might result in very complex local models.

3.2 Interaction Importance and Treewidth Based Variable-Cluster Augmentation

The balanced *variable-clusters* produced by the k-MinCut partitioning scheme are disjoint. Intuitively, there is significant correlation information that is lost during the partitioning. To compensate for this loss, we propose an interaction importance based *variable-cluster* augmenting scheme to recover the damaged correlation information. The idea is that for each *variable-cluster*, we let it grow outward. More specifically, it attracts and absorbs most significant (important) interactions (edges) incident to its vertices from outside to itself. As a result, some extra variables are pulled into the *variable-cluster*. We control the augmentation through the number of extra vertices pulled into the cluster (called *growth factor*). One can use the same growth factor for all *variable-clusters* to preserve their balance.

As an optimization, we account for the model complexity during the augmentation. We keep augmenting a partition until its complexity reaches a user-specified threshold.

More specifically, we keep track of the growth of the treewidth during the augmenting process for this purpose. 1-hop neighboring vertices are first considered by the augmentation, followed by 2-hop neighboring vertices and so on. Meanwhile, we still stick to the interaction importance criteria. The resultant augmented partitions are likely to become unbalanced in terms of their size. The partitions with a small treewidth will grow more significantly than those with a large treewidth. However, these partitions are balanced in terms of their complexity. A benefit of this scheme is that usually more interactions across different partitions will be accounted for in a computationally controllable manner, leading to a more accurate global model. Figure **??** presents a sketch of the overlapped *variable-clusters* after the augmentation.

3.3 Approximate Global MRFs and a Greedy Inference Algorithm

For each augmented *variable-cluster*, we collect all of its related itemsets and use the iterative scaling algorithm to learn an exact local model. This is computationally feasible since the local model corresponding to each *variable-cluster* is much simpler than the original model. Two local models are correlated to each other if they share variables. The collection of all local models forms a global model of the original transaction data. We note that this global model is an approximation of the exact global MRF, since we lose dependency information by breaking edges in the exact graphical model. However, most strong correlations are compensated during the *variable-cluster* augmentation. As such, we believe that the proposed global model reasonably approximates the exact model. Figure 2 provides the formal algorithm for learning an approximate global MRF.

LearnMRF(F, k, g)
Input : F, *collection of frequent itemsets*;
k, *number of partitions for MinCut partitioning*;
g, *growth factor*;
Output : $\mathcal{M}$, *global MRF*;
1. *Construct a weighted graph* G *from* F;
//G *specifies graphical structure of the exact MRF*;
2. k-*MinCut* G;
3. **for** *each graph partition* G_i
4. $G_i' \leftarrow augment(G_i, g)$;
5. *Pick itemsets* F_i *related to* G_i';
6. $M_i \leftarrow LearnLocalMRF(F_i)$;
7. *add* M_i *to* $\mathcal{M}$;
8. *return* $\mathcal{M}$;

Fig. 2. Learning approximate global MRF algorithm

Given the global model consisting of a set of local MRFs, how do we make inferences on this model efficiently? In the first case, where all query variables are subsumed by a single local MRF, we just need to calculate the marginal probability within the local model. In the second case, where query variables span multiple local models, we use a greedy decomposition scheme to compute. First, we pick the local model that has the largest intersection with the current query (i.e., covers most query variables). Then we pick the next local model that covers most uncovered variables in the query. This covering process will be repeated until we cover all variables in the query. Simultaneously,

all intersections between the above local models and the query are recorded. In the end, we derive an overlapped decomposition of the query. We notice that locally the dependency among small pieces in the decomposition often exhibits a tree-like structure, and we use Lemma 2 to compute the marginal probabilities.

Lemma 2. [3] *Given an undirected graph G subdivided into n overlapped components, if there exists an enumeration of these n components, i.e., C_1, C_2, ..., C_n, s.t., for any $2 \leq i \leq n$, the separating set, $s(C_i, \cup_{j=1}^{i-1} C_j) \subseteq (C_i \cap (\cup_{j=1}^{i-1} C_j))$, then the probability distribution associated with G is given by: $p(X) = \frac{\prod_{i=1}^{n} p(X_{C_i})}{\prod_{i=2}^{n} p(X_{C_i \cap (\cup_{j=1}^{i-1} X_{C_j})})}$*

Essentially, Lemma 2 specifies a junction tree-like structure. Given any model and one of its such decomposition, we can use the above formula to make exact inferences. However, it is possible to have cyclic dependencies among the decomposed pieces. Therefore, the greedy inference scheme is a heuristic. Also, we note that our global model is not globally consistent in that there exists inconsistency across the local models. However, we expect that the global model is *nearly consistent* since two correlated local models support the same evidence (itemsets) regarding their shared variables.

4 Experimental Results

In this section, we examine the performance of our proposed approach on real datasets. We focus on its application on the selectivity estimation problem. We compare our proposed model against the previous approach in [3] where a local MRF over query variables is learned for every query in an online fashion. We call this approach online local MRF approach (abbreviated as *OLM*).

Experimental Setup: All the experiments were conducted on a Pentium 4 2.66GHz machine with 1GB RAM running Linux 2.6.8. The MRF learning algorithm was implemented in C++. We used *apriori* [14] to collect frequent itemsets and *Metis* [12] to obtain a k-MinCut of the exact graphical model.
Datasets: We used two publicly available datasets in our experiments: the Microsoft Anonymous Web dataset (kdd.ics.uci.edu) with 32711 transactions and 294 items; the BMS-Webview1 dataset (fimi.cs.helsinki.fi) with 59602 transactions and 497 items.
Query Workloads: We considered the workloads consisting of conjunctive queries of different sizes. Following the same practice in [3], we first specified the number of query variables n (varied from 4, 6, 8, 10 to 12), then we picked n variables according to the probability of the variable taking a value of "1" and generated a value for each selected variable by its univariate probability distribution. **Performance Metrics:** We considered the *online time* cost, the time taken to answer the queries using the model. We also considered the *offline time* cost, the time taken to learn the model. We quantified the *accuracy* of estimations using the *average absolute relative error* over all queries in the workload. The absolute relative error is defined as $|\sigma - \hat{\sigma}| \,/\, \sigma$, where σ is the true selectivity and $\hat{\sigma}$ is the estimated selectivity.

[3] The complete proof can be found in the full version of this paper [13].

Results on the Microsoft Web Data. In this section, we report the experimental results on the Microsoft Web Data. We use the support threshold of 20 to collect the frequent itemsets, which results in 9901 frequent itemsets. According to the *Maximum Cardinality Search* (MCS)-ordering heuristic [15], the treewidth of the resulting MRF is 28 for which learning the exact model is intractable.

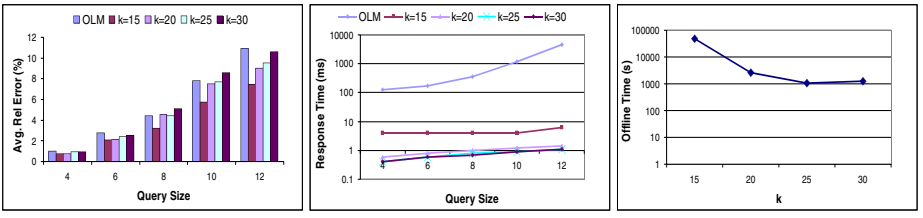


Fig. 3. Varying k ($g = 5$): (a) estimation accuracy (b) online time (c) offline time

Figure 3a presents the estimation accuracy when k is varied (g is fixed as 5) for queries of different sizes. As seen, our approach works very well compared with the online local MRF approach. Our approach gives very close or even better estimations compared with the online local MRF approach. These results are not surprising since for the online local model, we only use the local information to estimate the selectivity. However, for the offline global model, we rely on the global information to make the estimation. Even though the graph partitioning phase results in information loss, since the model is global in nature, in many cases it is still able to yield better estimations. Furthermore, an obvious trend that stands out is that as the query size increases, the quality of the estimations degrades. This is expected since for larger sized queries, estimation errors grow for both approaches. Another observation is that the estimations are more accurate when we use less *variable-clusters*. This is because with less *variable-clusters*, the information loss due to the graph partitioning is smaller, thus we capture better the correlations between partitions. Figure 3b illustrates how the online times depend on k. It can be clearly seen the significant growth of the online times taken by the online model (note the Y-axis scale). The extreme online timing efficiency of the offline model can be clearly seen from the results. In most cases, it outperformed the online model by two to three orders of magnitude. Further, we see that the smaller k results in higher online estimation time. This is expected since the smaller k is, the larger each local model will be, which explains the slower estimation. In the extreme case where k is 1, we revert to learning the exact global MRF, which has been shown to be computationally infeasible. Figure 3c presents the offline learning times of the offline model when varying k. An obvious trend is that as we increase k, overall the learning cost of the offline model decreases significantly. This is expected since the larger k results in less complex local models.

Figure 4a presents the estimation accuracy when varying g (k is fixed as 20). As one can see, the error decreases steadily with increasing g. When g is 0 (disjoint *variable-clusters*), the estimations are most inaccurate. In contrast, the estimations are much

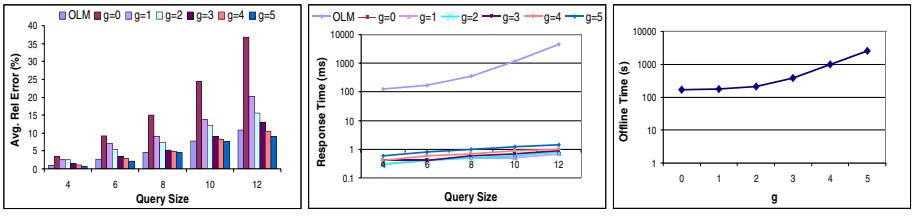


Fig. 4. Varying g ($k = 20$): (a) estimation accuracy (b) online time (c) offline time

more accurate when g is 5. The results clearly show the effects of the interaction importance based *variable-cluster* augmenting scheme. The offline model approximates the exact global model better when more correlations across the local models are compensated. Figure 4b presents the online times when varying g. We see from the results that the model with the larger g takes more online time to answer the query. This is also expected since the larger g results in more complex models (similar to the case of the smaller k). Figure 4c presents the offline learning times of the offline model when varying g. An obvious trend is that as we increase g, the time cost increases significantly, which is again expected.

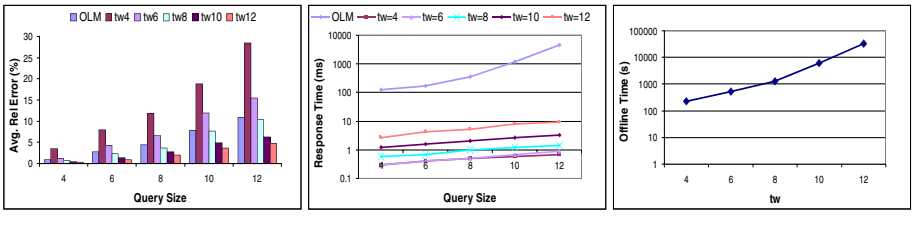


Fig. 5. Varying tw ($k = 25$): (a) estimation accuracy (b) online time (c) offline time

Figure 5a-c present the estimation accuracy, the online times and the offline learning times of the offline global model when the treewidth based augmentation optimization is used (k is fixed as 25). As seen, the optimization can further boost the estimation performance. For example, the average relative estimation errors are 0.29%, 0.97%, 2.01%, 3.66% and 4.81% on the workloads consisting of queries of size 4, 6, 8, 10 and 12, respectively. In contrast, the corresponding errors of the online local MRF approach are 0.99%, 2.76%, 4.45%, 7.82% and 10.9%, respectively. Furthermore, the offline model is faster by about two orders of magnitude in terms of online estimating time. Moreover, as we raise the treewidth threshold, the estimations will become more accurate, at a higher cost of online estimating and offline learning times.

The results on the BMS-Webview1 dataset overall are quite similar to that on the Microsoft Web dataset and are omitted in the interest of space. However, the complete results can be found in [13].

5 Conclusion

In this paper, we have described a new approach to learning an approximate MRF on large transaction data. Our proposed approach has been shown to be very effective and efficient in solving the selectivity estimation problem. In the future, we would like to exploit a belief propagation style approach to force the consistency of the model. Furthermore, we would like to investigate the use of the approximate inference techniques during the model learning process. Finally, it would be interesting to exploit the learned models on various link analysis tasks.

References

1. Getoor, L., Taskar, B., Koller, D.: Selectivity estimation using probabilistic models. In: SIGMOD Conference 2001. (2001) 461–472
2. Deshpande, A., Garofalakis, M.N., Rastogi, R.: Independence is good: Dependency-based histogram synopses for high-dimensional data. In: SIGMOD Conference 2001. (2001) 199–210
3. Pavlov, D., Mannila, H., Smyth, P.: Beyond independence: probabilistic models for query approximation on binary transaction data. IEEE Transactions on Knowledge and Data Engineering **15** (2003) 1409–1421
4. Breese, J.S., Heckerman, D., Kadie, C.M.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. (1998) 43–52
5. Goldenberg, A., Moore, A.: Tractable learning of large bayes net structures from sparse data. In: Proceedings of the twenty-first international conference on Machine learning. (2004)
6. Friedman, N.: Inferring cellular networks using probabilistic graphical models. Science **303** (2004) 799–805
7. Lauritzen, S., Speigelhalter, D.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, Series B (Methodological) **50** (1988) 157224
8. Jordan, M.I., Kearns, M.J., Solla, S.A.: An introduction to variational methods for graphical models. Machine Learning **37** (1999) 183–233
9. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. In: IJCAI. (2001)
10. Hollmen, J., Seppanen, J.K., Mannila, H.: Mixture models and frequent sets: Combining global and local methods for 0-1 data. In: Proceedings of the Third SIAM International Conference on Data Mining. (2003)
11. Jelinek, F.: Statistical Methods for Speech Recognition. MIT Press, Cambridge, MA (1998)
12. Karypis, G., Kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. J. Parallel Distrib. Comput. **48** (1998) 96–129
13. Wang, C., Parthasarathy, S.: Learning approximate mrfs from large transaction data. In: The Ohio State University, Technical Report. (2006)
14. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases. (1994) 487–499
15. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM Journal of Computing **13** (1984)

Similarity Search for Multi-dimensional NMR-Spectra of Natural Products

Karina Wolfram[1], Andrea Porzel[2], and Alexander Hinneburg[1]

[1] Institute of Computer Science
Martin-Luther-University of Halle-Wittenberg, Germany
{wolfram, hinneburg}@informatik.uni-halle.de
[2] Leibniz Institute of Plant Biochemistry (IPB), Germany
aporzel@ipb-halle.de

Abstract. Searching and mining nuclear magnetic resonance (NMR)-spectra of naturally occurring products is an important task to investigate new potentially useful chemical compounds. We develop a set-based similarity function, which, however, does not sufficiently capture more abstract aspects of similarity. NMR-spectra are like documents, but consists of continuous multi-dimensional points instead of words. Probabilistic semantic indexing (PLSI) is an retrieval method, which learns hidden topics. We develop several mappings from continuous NMR-spectra to discrete text-like data. The new mappings include redundancies into the discrete data, which proofs helpful for the PLSI-model used afterwards. Our experiments show that PLSI, which is designed for text data created by humans, can effectively handle the mapped NMR-data originating from natural products. Additionally, PLSI combined with the new mappings is able to find meaningful "topics" in the NMR-data.

1 Introduction

Nuclear magnetic resonance (NMR)-spectra are an important finger printing method to investigate the chemical structure of organic compounds from plants or other tissues. Two-dimensional-NMR spectroscopy is able to capture the influences of two different atom types at the same time (e.g. ^{1}H, hydrogen and ^{13}C carbon). The result of an 2D-NMR experiment can be seen as an intensity function measured over two variables[1]. Regions of high intensity are called peaks, which contain the real information about the underlying molecular structure. The usual visualizations of 2D-NMR spectra are contour plots as shown in figure 1. An ideal peak would register as a small dot, however, due to the limited resolution available (dependent on the strength of the magnetic field) multiple peaks may appear as a single merged object with non-convex shape. In the literature peaks are noted by their two-dimensional positions without any information about the shapes of the peaks. Content-based similarity search of 2D-NMR spectra would be a valuable tool for structure investigation by comparing spectra of unknown compounds with a set of spectra, for which the structures

[1] The measurements are in parts per million (ppm).

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 650–658, 2006.
© Springer-Verlag Berlin Heidelberg 2006

are known. While the principle is already in use for 1D-NMR spectra [5,4,1], to the best of our knowledge, no effective similarity search method is known for 2D-NMR-spectra.

Simplified, a 2D-NMR spectrum is a set of two-dimensional points. There is an analogy to text retrieval, where documents are usually represented as sets of words. Latent space models [3,2] were successfully used to model documents and thus improved the quality of text retrieval.

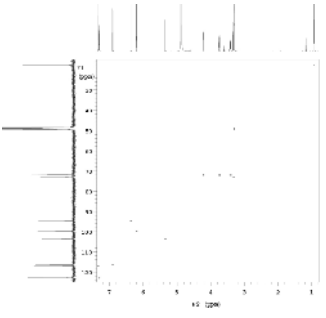


Fig. 1. 2D-NMR spectrum of quercetrin. The plots at the axes are the corresponding 1D-NMR spectra.

The contribution of this paper are methods to map 2D-NMR spectra to discrete text-like data, which can be analyzed and searched by any text retrieval method. Additionally, we propose a simple similarity function, which operates directly on the peaks of the spectra and serves as bottom line benchmark in the experimental evaluation. We demonstrate on real data that our mapping methods in combination with PLSI [3] improve the quality of similarity search of 2D-NMR spectra. Our results indicate at a larger scope that text retrieval and mining methods, designed for text data created by humans, in combination with appropriate mapping functions may yield the potential to be also successful for experimental data from naturally occurring objects. In this paper we consider exemplarily ^{1}H, ^{13}C one-bond heteronuclear shift correlation 2D-NMR spectra.

The paper is structured as follows: first, in section 2, we define 2D-NMR spectra and propose a simple similarity function. In section 3, we propose the new mapping functions for 2D-NMR spectra. In section 4, we describe our experimental evaluation and section 5 concludes the paper.

2 Directly Computing Similarity

A two-dimensional NMR-spectrum of an organic compound captures many structural characteristics like rings and chains. Most important are the positions of the peaks. As the shape of a peak and its height (intensity) strongly varies over different experiments with the same compound, the representation of a spectrum includes the peak positions only. A **2D NMR-spectrum** A is defined as a set of points $\{x_1, \ldots, x_n\} \subset \mathbb{R}^2$. The $|\cdot|$ function denotes the size of the spectrum $|A| = n$. A peak matches other peaks only within a certain spatial neighborhood, which is defined by the ranges α and β. A peak x from spectrum A **matches** a peak y from spectrum B, if $|x.c - y.c| < \alpha$ and $|x.h - y.h| < \beta$, where $.c$ and $.h$ denote the NMR measurements for carbon and hydrogen respectively. Note that a single peak of a spectrum can match several peaks from another spectrum. Given two spectra A and B, the subset of peaks from A which find matching partners in B is denoted as $matches(A, B) = \{x\colon x \in A, \exists y \in B\colon x \text{ matches } y\}$.

The function *matches* is not symmetric, but helps to define a symmetric similarity measure Let be A and B two spectra and $A' = matches(A, B)$ and $B' = matches(B, A)$, so the **similarity** is defined as

$$sim(A, B) = \frac{|A'| + |B'|}{|A| + |B|}$$

The measure is close to one if most peaks of both spectra are matching peaks. Otherwise the similarity drops towards zero.

3 Mapping 2D-NMR-Spectra to Document-Like Data

Like a 2D-NMR spectrum consists of a set of peaks, a document consists of many words, which typically are modeled as a set. So assuming a 2D-NMR spectrum can be transformed into a text-like object by mapping the continuous 2D peaks to discrete variables, a variety of text retrieval models can be applied. However, it is an open question, whether models designed for quite different data, namely texts created by humans, are effective on data which comes for naturally occurring compounds and thus do not include human design patterns. Because the patterns which are important to 2D-NMR spectra similarity search might be quite different from pattern found in document collections, we chose a retrieval model which is capable of learning relevant patterns from training data. Probabilistic latent semantic indexing (PLSI) introduced in [3] is a model for text retrieval with such a learning ability. For 2D-NMR spectra similarity search it is not clear, what is the best way to map the peaks of a spectrum to discrete words.

In this section we propose different methods to map the peaks of an NMR-spectrum from the continuous space of measurements to a discrete space of words. With the help of such a mapping, methods for text retrieval like PLSI can be directly applied. However, the quality of the similarity search depend on how the peaks are mapped to discrete words.

3.1 Grid-Based Mapping

First, we introduce a simple grid-based method, on which we will build more sophisticated methods. A simple grid-based method is to partition each of the both axis of the two-dimensional peak space into intervals of same size. Thus, an equidistant grid is induced in the two-dimensional peak space and a peak is mapped to exactly one grid cell it belongs to. When a grid cell is identified by a discrete integer vector consisting of the cells coordinates the mapping of a peak $x \in \mathbb{R}^2$ is formalized as $g(x) = (g_c(x.c), g_h(x.h))$ with $g_c(x.c) = \left\lfloor \frac{x.c}{w_c} \right\rfloor$, $g_h(x.h) = \left\lfloor \frac{x.h}{w_h} \right\rfloor$ The quantities w_c and w_h are the extensions of a cell in the respective dimensions, which are parameters of the mapping. The grid is

centered at the origin of the peak space. The cells of the grid act as words. The vocabulary generated by the mapped peaks consists of those grid cells which contain at least one peak. Empty grid cells are not included in the vocabulary. A word consists of a two-dimensional discrete integer vector.

Unfortunately the grid-based mapping has two disadvantages. First, close peaks may be mapped to different grid cells. This may lead to poor matching of related peaks in the discrete word space. Second, peaks of new query spectra are ignored when they are mapped to grid cells not included in the vocabulary. So some information from the query is not used for the similarity search which may weaken the performance.

3.2 Redundant Mappings

We propose three mappings which introduce certain redundancies by mapping a single peak to a set of grid cells. The redundancy in the new mappings shall compensate for the drawbacks of the simple grid-based mapping.

Shifted Grids. The first disadvantage of the simple grid-based method is that peaks which are very close in the peak space may be mapped to different grid cells, because a cell border is between them. So proximity of peaks does not guaranty that they are mapped to the same discrete cell.

Instead of mapping a peak to a single grid cell, we propose to map it to a set of overlapping grid cells. This is achieved by several shifted grids of the same granularity. In addition to the base grid some grids are shifted into the three directions $(1,0)(0,1)(1,1)$. One grid is shifted in each of the directions by half of the extent of a cell. In general, there may be $k-1$ grids shifted by fractions of $1/k, 2/k, \ldots, k-1/k$ of the extent of a cell in each direction respectively. For the mapping of the peaks to words which consist of cells from the different grids, two additional dimensions are needed to distinguish (a) the $k-1$ grids in each direction and (b) the directions themselves. The third coordinate represents the fraction by which a cell is shifted and the fourth one represents the directions by the following coding: value 0 is (0,0), 1 is (1,0), 2 is (0,1) and 3 is (1,1). So each peak is mapped to a finite set of four-dimensional integer vectors. The mapping of a peak $x \in \mathbb{R}^2$ is

$$s(x) = \{(g_c(x.c), g_h(x.h), 0, 0)\} \cup \bigcup_{i=1}^{k-1} \big\{(g_c(x.c + i/k \cdot w_c), g_h(x.h), i, 1),$$
$$(g_c(x.c), g_h(x.h + i/k \cdot w_h), i, 2), (g_c(x.c + i/k \cdot w_c), g_h(x.h + i/k \cdot w_h), i, 3)\big\}$$

Thus, a single peak is mapped to $3(k-1)+1$ words. A nice property of the mapping is that there exists at least one grid cell for every pair of matching peaks both peaks are mapped to.

Different Resolutions. The second disadvantage of the simple grid-based mapping comes from the fact that empty grid cells (not occupied by at least

one peak from the set of training spectra) do not contribute to the representation to be learned for similarity search. So peaks of new query spectra mapped to those empty cells are ignored. That effect can be diminished by making the grid cells larger. However, this is counterproductive for the precision of the similarity search due to the coarser resolution. Thus, there are two contradicting goals, namely (a) to have a fine resolution to handle subtle aspects in the data and (b) to cover at the same time the whole peak space by a coarse resolution grid so that no peaks of a new query spectrum have to be ignored.

Instead of finding a tradeoff for a single grid, both goals can be served by combining simple grids with different resolutions. Given l different resolutions $\{(w_c^{(1)}, w_h^{(1)}), \ldots, (w_c^{(l)}, w_h^{(l)})\}$ a peak is mapped to l grid cells of different sizes. In order to distinguish between the different grids an additional discrete dimension is needed. So the mapping function is

$$r(x) = \bigcup_{i=1}^{l} \{(g_c^{(i)}(x), g_h^{(i)}(x), i)\}$$

with $g_c^{(i)}$ and $g_h^{(i)}$ use $w_c^{(i)}$ and $w_h^{(i)}$ respectively. Note that a hierarchical, quadtree like partitioning is a special case of the proposed mapping function with $w_c^{(i)} = 2^{i-1} w_c$ and $w_h^{(i)} = 2^{i-1} w_h$.

Combining shifted Grids with different Resolutions. Both methods are designed to compensate for different drawbacks of the simple grid mapping. So it is natural to combine both mappings. The parameters of such a mapping are the number of shifts k, the number of different grid cell sizes l and the actual sizes $\{(w_c^{(1)}, w_h^{(1)}), \ldots, (w_c^{(l)}, w_h^{(l)})\}$. Beside the two coordinates for the grid cells, additional discrete dimensions are needed for the shift, the direction and the grid resolution. Using the the definitions from above the mapping function of the combined mapping of a peak is

$$c(x) = \bigcup_{i=1}^{l} \{(g_c^{(i)}(x.c), g_h^{(i)}(x.h), 0, 0, i)\} \cup \bigcup_{j=1}^{k-1} \Big\{ (g_c^{(i)}(x.c + i/k \cdot w_c^{(i)}), g_h^{(i)}(x.h), j, 1, i),$$
$$(g_c^{(i)}(x.c), g_h^{(i)}(x.h + i/k w_h^{(i)}), j, 2, i), (g_c^{(i)}(x.c + i/k w_c^{(i)}), g_h^{(i)}(x.h + i/k \, w_h^{(i)}), j, 3, i) \Big\}$$

Thus a single peak is mapped to $l(3(k-1)+1)$ words. In the next section all mappings are compared with respect to the effectiveness for similarity search.

4 Evaluation and Results

The data used are mostly secondary metabolites of plants and fungi. The substances cover a representative area of naturally occurring compounds The database includes about 587 spectra, each has about 3 to 35 peaks. The total number of

Group	#Spectra	#Peaks
Pregnans	11	17–26
Anthrquinones	8	3–6
Aconitanes	8	22–26
Triterpenes	17	24–31
Flavonoids	18	5–8
Isoflavonoids	16	5–7
Aflatoxins	8	8–10
Steroids	12	16–23
Cardenolides	15	18–25
Coumarins	19	3–8

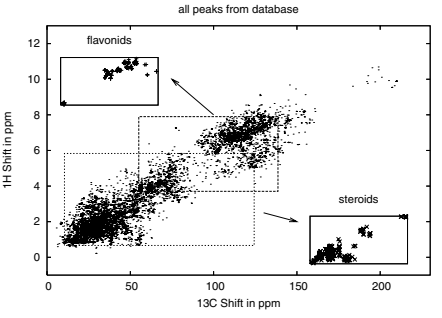


Fig. 2. Left: Groups with number of spectra and range of peaks, Right: Distribution of the peaks of all spectra with the distribution within the groups of flavonoids and steroids

peaks is 7029. Ten small groups of chemically similar compounds are included in the database for controlled experiments. The groups with the number of spectra and number of peaks are listed in figure 2 left. The peak space with all peaks in the database is shown in figure 2 right.

4.1 Comparison

The different methods for similarity search of 2D-NMR-spectra are compared using recall-precision curves. The search quality is high, when both – recall and precision – are high. So the upper curves are the best.

First, the direct similarity function is tested. Each spectrum from the ten groups is used as a query while the rest of the respective group should be found as answers. The plot in figure 3a shows averages over all queries. The size of the matching neighborhood is varied over $\alpha = 4, 6, 8, 10$ and $\beta = 0.4, 0.6, 0.8, 1.0$ respectively. As expected, the search quality is low. In fact on average, it fails to deliver a spectrum from the answer set in the top ranks which is indicated by the hill-like shape of the curves.

Next, a series of experiments is conducted using our proposed mapping functions in combination with PLSI. All curves are averages from cross validation over all groups. As the groups are very small the leave-one-out testing scheme is employed. The results for the simple grid-based mapping are shown in figure 3b. The sizes of the grid cells are varied over $w_c = 4, 6, 8, 10$ and $w_h = 0.4, 0.6, 0.8, 1.0$ respectively. The results are already much better than those for the direct similarity function. Small sizes give the best results. The use of shifted grids improves the performance substantially over simple grids, as shown in figure 3c,d. The plots show the experiments for $k = 2, 3$. The quality of $w_c = 4$ and $w_h = 0.4$ with $k = 2$ and $k = 3$ are almost identical. However, the vocabulary for $k = 2$ is much smaller, so the model has much less parameters to train. In practise, the smaller model with $k = 2$ shifts is favored.

Also the mapping based on grids with different grid cell sizes are assessed. Due to lack of space, only the results from combinations of $w_c^{(1)} = 4, w_h^{(1)} = 0.4$ with

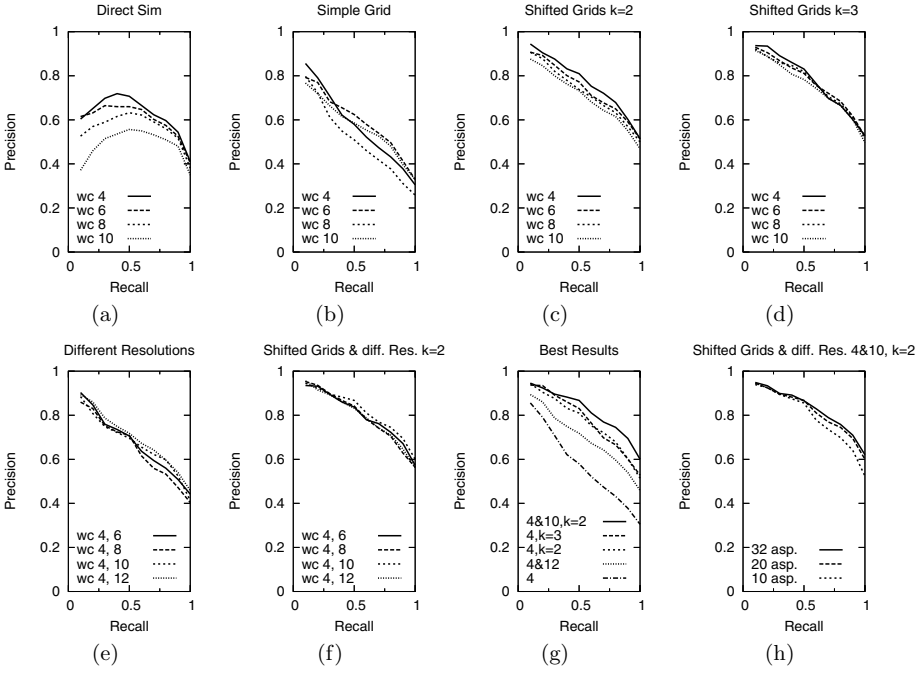


Fig. 3. Average recall-precision curves from leave-one-out cross validation experiments

other sizes are reported, because those performed best among all combinations. Figure 3e shows that also the mapping based on different grid cell sizes outperforms the simple grid-based mapping. But the improvement is not as much as for shifted grids. The set of resolutions $\{(w_c^{(1)} = 4, w_h^{(1)} = 0.4), (w_c^{(2)} = 12, w_h^{(2)} = 1.2)\}$ performs best.

Last, experiments are performed with the combination of the previous two mappings, namely a combination of shifted grids with those of different resolutions. The performance results are shown in figure 3f which indicates that the best combination, namely the resolution set $\{(w_c^{(1)} = 4, w_h^{(1)} = 0.4), (w_c^{(2)} = 10, w_h^{(2)} = 1.0)\}$ with $k = 2$ shifts, outperforms both previous mappings. This is more clearly seen in figure 3g which compares the best performing settings from the above experiments. In summery, the mappings based on shifted grids and those with different resolutions perform significantly better than the simple grid-based mapping. Finally, the combination of shifted grids and grids with different resolutions is even better than the individual mappings.

The last point is the number of hidden aspects. For the experiments reported so far, the PLSI model is used with 20 hidden aspects. Also different numbers of aspects are tested using the best combination of mappings. Figure 3h shows that the performance with 10 aspects drops a bit The increase in the numbers of aspects from 20 to 32 is only marginally reflected in increase of search performance. So 20 is a reasonable number of aspects for the given data. In

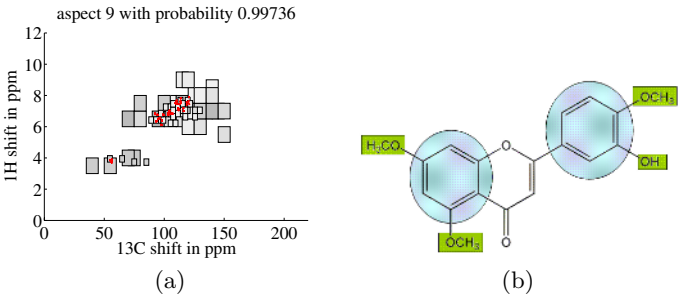


Fig. 4. (a) Main aspect of the flavonoid group which includes the region of aromatic rings (upper right cluster) and the region for oxygen substituents (lower left cluster). The gray shades indicate the strength of the association between grid cell and aspect. (b) An example of an flavonoid (3'-Hydroxy-5,7,4'-trimethoxyflavone) where the aromatic rings and the oxygen substituents (methoxy groups in this case) are marked.

conclusion, the results prove experimentally that the PLSI model, designed for text retrieval, is indeed effective for similarity search of 2D-NMR spectra from naturally occurring compounds.

4.2 Analysis of the latent Aspects

We analyzed the latent aspects learned by the PLSI model using the mapping based on the combination of shifted grids with different resolutions. The grid cells (words) with high probability for a given aspect are plotted together to describe the aspects meaning. Some aspects specialized on certain regions in the peak space which are typical for distinct molecule fragments like aromatic rings or alkane skeletons. However, also more subtle details of the data are captured by the aspect model. For example, the main aspect for the group of flavonoids specializes not only on the region for aromatic rings which are the main part of flavonoids. It also includes a smaller region which indicates oxygen substitution. A closer inspection of the database revealed that indeed many of the included flavonoids do have several oxygen substituents. The main aspect for flavonoids with the respective peak distribution of the flavonoid group is shown in figure 4a. We believe a detailed analysis of the aspects found by the model may help to investigate unknown structures of new substances when their NMR-spectra are included in the training set.

5 Conclusion

We proposed redundant mappings from continuous 2D-NMR spectra to discrete text-like data which can be processed by any text retrieval method. We demonstrated experimentally the effectiveness of the our mappings in combination with PLSI. Further analysis revealed that the aspects found by PLSI are chemically relevant. In future research we will study more recent text models like LDA [2] in combination with our mapping methods.

References

1. A. S. Barros and D. N. Rutledge. Segmented principal component transform-principal component analysis. *Chemometrics & Intelligent Laboratory Systems*, 78:125–137, 2005.
2. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
3. T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99*, 1999.
4. P. Krishnan, N. J. Kruger, and R. G. Ratcliffe. Metabolite fingerprinting and profiling in plants using nmr. *Journal of Experimental Botany*, 56:255–265, 2005.
5. C. Steinbeck, S. Krause, and S. Kuhn. Nmrshiftdb-constructing a free chemical information system with open-source components. *J. chem. inf. & comp. sci.*, 43: 1733 –1739, 2003.

Author Index