Peter A. Flach
Tijl De Bie
Nello Cristianini (Eds.)

LNAI 7524

Machine Learning and Knowledge Discovery in Databases

European Conference, ECML PKDD 2012
Bristol, UK, September 2012
Proceedings, Part II

2 Part II

Springer

Lecture Notes in Artificial Intelligence 7524

Subseries of Lecture Notes in Computer Science

Peter A. Flach Tijl De Bie
Nello Cristianini (Eds.)

Machine Learning and Knowledge Discovery in Databases

European Conference, ECML PKDD 2012
Bristol, UK, September 24-28, 2012
Proceedings, Part II

Springer

Volume Editors

Peter A. Flach
Tijl De Bie
Nello Cristianini
University of Bristol
Intelligent Systems Laboratory
Merchant Venturers Building
Woodland Road
Bristol BS8 1UB, UK
E-mails:
peter.flach@bristol.ac.uk
tijl.debie@bristol.ac.uk
nello.cristianini@bristol.ac.uk

© Cover illustration by www.zedphoto.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-33485-6 e-ISBN 978-3-642-33486-3
DOI 10.1007/978-3-642-33486-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: Applied for

CR Subject Classification (1998):
I.2.6, H.2.8, I.5.2, G.2.2, G.3, I.2.4, I.2.7, H.3.4-5, I.2.9, F.2.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

These proceedings contain the technical papers presented at the 2012 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2012), held in Bristol, UK, during the week of 24–28 September 2012. ECML-PKDD is a highly successful and selective international conference series, which was first organised in its present form in 2001 in Freiburg, Germany, when it joined together the hitherto separate ECML and PKDD conferences. Since then, the two strands of machine learning and data mining have been increasingly integrated in the joint conference, and today it is well-known as the only top-tier international conference that exploits the synergy between these two exciting fields. It is therefore particularly pleasing that the achieved level of synergy is evident from the list of topics in which the research track papers are categorised in these pages: Association Rules and Frequent Patterns; Bayesian Learning and Graphical Models; Classification; Dimensionality Reduction, Feature Selection and Extraction; Distance-Based Methods and Kernels; Ensemble Methods; Graph and Tree Mining; Large-Scale, Distributed and Parallel Mining and Learning; Multi-relational Mining and Learning; Multi-task Learning; Natural Language Processing; Online Learning and Data Streams; Privacy and Security; Rankings and Recommendations; Reinforcement Learning and Planning; Rule Mining and Subgroup Discovery; Semi-supervised and Transductive Learning; Sensor Data; Sequence and String Mining; Social Network Mining; Spatial and Geographical Data Mining; Statistical Methods and Evaluation; Time Series and Temporal Data Mining; and Transfer Learning.

The format of the 2012 conference follows the tried-and-tested format of previous instalments, with workshops and tutorials on Monday and Friday; research papers in parallel tracks on Tuesday, Wednesday and Thursday; and plenary keynote talks on each of the five conference days. The proceedings start with abstracts and bios of our five eminent invited speakers: Pieter Abbeel, Luc De Raedt, Douglas Eck, Daniel Keim and Padhraic Smyth. The bulk of the proceedings is then taken up by 105 research papers. These were carefully selected from 443 submitted papers (acceptance rate 23.7%) on the basis of reviews by 275 Programme Committee members and 36 Area Chairs, assisted by 161 additional reviewers. In acknowledgment of their essential contribution to the success of the conference you will find their names on the following pages.

The final sections of the proceedings are devoted to Demo and Nectar papers. Ten system demonstration papers were selected from 19 submissions to the Demo track by a small committee chaired by Bettina Berendt and Myra Spiliopoulou. The Nectar track is new this year and features significant machine learning and data mining results published or disseminated no earlier than 2010 at different conferences or in journals. One goal of this track is to offer conference attendees the opportunity to learn about results published in other communities but related to machine learning or data mining (or both). Submissions compactly presenting well-founded results which appeared in a series of publications that advanced a single novel influential idea or vision were also welcomed. A dedicated committee chaired by Gemma Garriga and Thomas Gärtner selected 4 Nectar papers from 14 submissions. Our sincere thanks to everyone involved for these valuable additions to the conference.

Elements of the conference not directly represented in the proceedings include 11 workshops (Mining Ubiquitous and Social Environments; New Frontiers in Mining Complex Patterns; The Silver Lining – Learning from Unexpected Results; Instant Interactive Data Mining; Learning and Discovery in Symbolic Systems Biology; Sentiment Discovery from Affective Data; Active Learning in Real-world Applications; Mining and Exploiting Interpretable Local Patterns; Community Mining and People Recommenders; Collective Learning and Inference on Structured Data: and the Discovery Challenge workshop), as well as 8 tutorials (Understanding and Managing Cascades on Large Graphs; Advanced Topics in Data Stream Mining; Mining Deep Web Repositories; PAC-Bayesian Analysis in Supervised, Unsupervised, and Reinforcement Learning; Random Projections for Machine Learning and Data Mining; Decomposing Binary Matrices; Advanced Topics in Ensemble Learning; and Probabilistic Modeling of Ranking). Many thanks to the workshop organisers and tutorial presenters, as well as the Workshop Chairs Arno Knobbe and Carlos Soares and the Tutorial Chairs Alessandro Moschitti and Siegfried Nijssen for putting together this exciting programme. We would also like to draw attention to the programme of presentations by representatives from industry put together by Industry Track Chairs Cédric Archambeau and David Barber, consisting of a series of talks centred around Big Data as well as a programme of 'Startup Stories' sponsored by the PASCAL2 Network of Excellence.

Finally, it is our pleasure to announce the winners of the best paper awards, as selected by a small committee chaired by the Awards Chair Johannes Fürnkranz. The paper 'Active Evaluation of Ranking Functions based on Graded Relevance' by Christoph Sawade, Steffen Bickel, Timo von Oertzen, Tobias Scheffer and Niels Landwehr wins the best machine learning paper award sponsored by the *Machine Learning* journal. The paper 'Socioscope: Spatio-Temporal Signal Recovery from Social Media' by Jun-Ming Xu, Aniruddha Bhargava, Robert Nowak and Xiaojin Zhu receives the best data mining paper award sponsored by *Data Mining and Knowledge Discovery.* Our congratulations to the authors of both papers and in particular to their student main authors. We also continue the tradition started at ECML-PKDD 2011 in Athens of selecting a most influential paper published at the conference 10 years ago. Following a poll organised by the Awards Chair among selected participants the most influential paper presented at ECML-PKDD 2002 in Helsinki is 'Mining All Non-derivable Frequent Itemsets' by Toon Calders and Bart Goethals. We look forward to finding out which paper from these proceedings will be deemed to have been most influential in 2022!

July 2012

Peter Flach
Tijl De Bie
Nello Cristianini

Organisation

ECML-PKDD 2012 was organised by the Departments of Computer Science and Engineering Mathematics of the University of Bristol, UK. We gratefully acknowledge the financial support of our academic sponsors (the University of Bristol (the aforementioned Departments as well as the Faculty of Engineering), the PASCAL2 Network of Excellence and the EternalS FP7 Coordination Action), our business sponsors (Winton Capital Management, Hewlett-Packard Research Labs Bristol, Rapid-I, Cambridge University Press, Google, KNIME, IBM UK and Microsoft Research) and our award sponsors (the journals of *Machine Learning* and *Data Mining and Knowledge Discovery*).

General and Programme Chairs

Peter Flach	University of Bristol, UK
Tijl De Bie	University of Bristol, UK
Nello Cristianini	University of Bristol, UK

Local Organisers

Oliver Ray	University of Bristol, UK
Tilo Burghardt	University of Bristol, UK
Nanlin Jin	University of Bristol, UK
Yizhao Ni	University of Bristol, UK
Simon Price	University of Bristol, UK

Workshop Chairs

Arno Knobbe	Universiteit Leiden, The Netherlands
Carlos Soares	Universidade do Porto, Portugal

Tutorial Chairs

Alessandro Moschitti	University of Trento, Italy
Siegfried Nijssen	Katholieke Universiteit Leuven, Belgium

Demo Track Chairs

Bettina Berendt	Katholieke Universiteit Leuven, Belgium
Myra Spiliopoulou	University of Magdeburg, Germany

Nectar Track Chairs

Gemma C. Garriga	INRIA Lille, France
Thomas Gärtner	University of Bonn & Fraunhofer, Germany

Industry Track Chairs

Cédric Archambeau	Xerox Research Centre Europe, France
David Barber	University College London, UK

Awards Chair

Johannes Fürnkranz	TU Darmstadt, Germany

Sponsorship Chairs

Toon Calders	Eindhoven University, The Netherlands
Trevor Martin	University of Bristol, UK

Publicity Chair

Grigorios Tsoumakas	Aristotle University of Thessaloniki, Greece

Proceedings Chairs

Ilias Flaounas	University of Bristol, UK
Tim Kovacs	University of Bristol, UK

Webmaster

Thomas Lansdall-Welfare	University of Bristol, UK

Discovery Challenge Organisers

Ion Androutsopoulos	Athens University of Economics and Business, Greece
Thierry Artieres	Laboratoire d'Informatique de Paris 6, France
Patrick Gallinari	Laboratoire d'Informatique de Paris 6, France
Eric Gaussier	Laboratoire d'Informatique de Grenoble, France
Aris Kosmopoulos	NCRS "Demokritos" & Athens University of Economics and Business, Greece
George Paliouras	NCRS "Demokritos", Greece
Ioannis Partalas	Laboratoire d'Informatique de Grenoble, France

ECML-PKDD Steering Committee

Fosca Giannotti, chair	Università di Pisa, Italy
Francesco Bonchi	Yahoo! Research Barcelona, Spain
Wray Buntine	NICTA Canberra, Australia
Dimitrios Gunopulos	University of Athens, Greece
Donato Malerba	Università degli Studi di Bari, Italy
Dunja Mladenić	Jožef Stefan Institute, Slovenia
John Shawe-Taylor	University College London, UK
Michèle Sebag	Laboratoire de Recherche en Informatique, France
Michalis Vazirgiannis	Athens University of Economics and Business, Greece

Area Chairs

Annalisa Appice	Università degli Studi di Bari, Italy
Roberto J. Bayardo	Google, USA
Tanya Berger-Wolf	University of Illinois, USA
Hendrik Blockeel	Katholieke Universiteit Leuven, Belgium
Francesco Bonchi	Yahoo! Research Barcelona, Spain
Carla E. Brodley	Tuft University, USA
Carlotta Domeniconi	George Mason University, USA
Tina Eliassi-Rad	Rutgers University, USA
Charles Elkan	University of California San Diego, USA
Tapio Elomaa	Tampere University of Technology, Finland
Wei Fan	IBM T.J.Watson Research, USA
Paolo Frasconi	Università degli Studi di Firenze, Italy
João Gama	University of Porto, Portugal
Gemma C. Garriga	INRIA Lille Nord Europe, France
Claudio Gentile	Università dell'Insubria, Italy
Aristides Gionis	Yahoo! Research Barcelona, Spain
Geoff Holmes	University of Waikato, New Zealand
Eyke Hüllermeier	Philipps-Universität Marburg, Germany
George Karypis	University of Minnesota, USA
Kristian Kersting	University of Bonn, Germany
Joost Kok	University of Leiden, the Netherlands
James Kwok	Hong Kong University of Science and Technology, China
Bing Liu	University of Illinois, USA
Marie-Francine Moens	Katholieke Universiteit Leuven, Belgium
Alessandro Moschitti	University of Trento, Italy
Mahesan Niranjan	University of Southampton, UK
Dino Pedreschi	Università di Pisa, Italy
Jian Pei	Simon Fraser University, Canada
Bernhard Pfahringer	University of Waikato, New Zealand
Teemu Roos	University of Helsinki, Finland
Arno Siebes	University of Utrecht, the Netherlands
Myra Spiliopoulou	University of Magdeburg, Germany

Hannu Toivonen	University of Helsinki, Finland
Luis Torgo	University of Porto, Portugal
Jean-Philippe Vert	Mines ParisTech & Curie Institute, France
Stefan Wrobel	University of Bonn & Fraunhofer, Germany

Programme Committee

Naoki Abe
Nitin Agarwal
Fabio Aiolli
Florence d'Alche-Buc
Aris Anagnostopoulos
Gennady Andrienko
Ana Paula Appel
Marta Arias
Ira Assent
Martin Atzmueller
Bart Baesens
José Balcázar
Elena Baralis
Shai Ben David
Bettina Berendt
Michele Berlingerio
Michael W. Berry
Michael R. Berthold
Albert Bifet
Misha Bilenko
Mustafa Bilgic
Paolo Boldi
Mario Boley
Christian Borgelt
Henrik Boström
Stephane Boucheron
Remco R. Bouckaert
Anne-Laure Boulesteix
Jean-Francois Boulicaut
Marc Boullé
Pavel Brazdil
Ulf Brefeld
Sebastien Bubeck
Wray Buntine
Tiberio Caetano
Deng Cai
Toon Calders
Colin Campbell
Stephane Canu
Olivier Cappé
Xavier Carreras
Andre Carvalho
Alexandra Carvalho
Michelangelo Ceci
Tania Cerquitelli
Hong Cheng
Weiwei Cheng
Jesús Cid-Sueiro
Frans Coenen
Fabrizio Costa
James Cussens
Alfredo Cuzzocrea
Jesse Davis
Krzysztof Dembczyński
Janez Demšar
Christian Desrosiers
Tom Diethe
Kurt Driessens
Chris Drummond
Pierre Dupont
Saso Dzeroski
Floriana Esposito
A. Fazel Famili
Nicola Fanizzi
Tom Fawcett
Ad Feelders
Xiaoli Z. Fern
Cèsar Ferri
Daan Fierens
Patrik Floréen
George Forman
Blaž Fortuna
Elisa Fromont
Johannes Fürnkranz
Mohamed Gaber
Thomas Gärtner
Brian Gallagher
Patrick Gallinari
Eric Gaussier
Ricard Gavaldà
Peter Geibel
Pierre Geurts
Mohammad Ghavamzadeh
Fosca Giannotti
Rémi Gilleron
Christophe G. Giraud-Carrier
Aris Gkoulalas-Divanis
Bart Goethals
Marco Gori
Henrik Grosskreutz
Steve R. Gunn
Stephan Günnemann
Dimitrios Gunopulos
Jiawei Han
Edwin Hancock
Mohammad Hasan
Mark Herbster
José Hernández-Orallo
Colin de la Higuera
Alexander Hinneburg
Frank Hoeppner
Jaakko Hollmén
Tamas Horvath
Andreas Hotho
Minqing Hu
Ming Hua
Ramon Huerta
Georgiana Ifrim
Nathalie Japkowicz
Matti Järvisalo
Alexandros Kalousis
Hilbert Kappen
Hillol Kargupta

Panagiotis Karras
Hisashi Kashima
Samuel Kaski
Latifur Khan
Marius Kloft
Mikko Koivisto
Tamara G. Kolda
Petri Kontkanen
Walter A. Kosters
Samory Kpotufe
Stefan Kramer
Shonali Krishnaswamy
Nicolas Lachiche
Nada Lavrač
Matthijs Van Leeuwen
Jiuyong Li
Tao Li
Xiaoli Li
Jefrey Lijffijt
Aristidis Likas
Charles Ling
Marco Lippi
Francesca A. Lisi
Xiaohui Liu
Corrado Loglisci
Daniel Lowd
Sofus A. Macskassy
Donato Malerba
Giuseppe Manco
Dragos D Margineantu
Stan Matwin
Dimitrios Mavroeidis
Michael May
Mike Mayo
Thorsten Meinl
Prem Melville
Ernestina Menasalvas
Aditya Menon
Rosa Meo
Pauli Miettinen
Dunja Mladenić
Katharina J. Morik
Klaus-Robert Muller
Emmanuel Müller
Ricardo Ñanculef
Olfa Nasraoui
Sriraam Natarajan
Jennifer Neville
Yizhao Ni
Siegfried Nijssen
Keith Noto
Andreas Nürnberger
Guillaume Obozinski
Francesco Orabona
Salvatore Orlando
Gerhard Paaß
Tapio Pahikkala
George Paliouras
Spiros Papadimitriou
Panagiotis Papapetrou
Emilio Parrado-Hernandez
Srinivasan Parthasarathy
Andrea Passerini
Mykola Pechenizkiy
Marcello Pelillo
Jaakko Peltonen
Pedro Pereira Rodrigues
Fernando Perez-Cruz
Gregory Piatetsky-Shapiro
Enric Plaza
Massimiliano Pontil
Ronaldo C. Prati
Kai Puolamäki
Chedy Raïssi
Alain Rakotomamonjy
Liva Ralaivola
Naren Ramakrishnan
Jan Ramon
Huzefa Rangwala
Balaraman Ravindran
Patricia Riddle
Fabrizio Riguzzi
Fabio Roli
Lorenzo Rosasco
Volker Roth
Juho Rousu
Céline Rouveirol
Cynthia Rudin
Ulrich Rueckert
Stefan Rüping
Maytal Saar-Tsechansky
Lorenza Saitta
Scott Sanner
Vítor Santos Costa
Raul Santos-Rodriguez
Tobias Scheffer
Lars Schmidt-Thieme
Dale Schuurmans
Michèle Sebag
Thomas Seidl
Ricardo Silva
Andrzej Skowron
Kevin Small
Carlos Soares
Richard Socher
Maarten van Someren
Mauro Sozio
Alessandro Sperduti
Masashi Sugiyama
Jimeng Sun
Johan Suykens
Einoshin Suzuki
Sandor Szedmak
Prasad Tadepalli
Andrea Tagarelli
Pang-Ning Tan
Lei Tang
Nikolaj Tatti
Evimaria Terzi
Ivan Titov
Ljupčo Todorovski
Hanghang Tong
Volker Tresp
Konstantin Tretyakov
Ivor W. Tsang
Panayiotis Tsaparas
Grigorios Tsoumakas
Koji Tsuda
Alan Tucker
Antti Ukkonen
Joaquin Vanschoren
Michalis Vazirgiannis
Shankar Vembu

Herna L. Viktor
Fabio Vitale
Christel Vrain
Jilles Vreeken
Willem Waegeman
Jianyong Wang
Hongning Wang
Liwei Wang
Wei Wang
Gerhard Widmer
Hui Xiong
Zhao Xu
Jieping Ye
Mi-Yen Yeh
Philip Yu
Gerson Zaverucha
Filip Železný
Dell Zhang
Kai Zhang
Lei Zhang
Min-Ling Zhang
Mark Zhang
Ying Zhao
Zheng Zhao
Zhi-Hua Zhou
Arthur Zimek
Indre Zliobaite
Jean-Daniel Zucker
Blaž Zupan

Demo Track Programme Committee

Omar Alonso
Steve Beitzel
Paul Bennett
Michael Berthold
Albert Bifet
Francesco Bonchi
Christian Borgelt
Jaakko Hollmén
Arvind Hulgeri
Ralf Klinkenberg
Michael Mampaey
Michael May
Gabor Melli
Gerard de Melo
Rosa Meo
Themis Palpanas
Mykola Pechenizkiy
Peter van der Putten
Daniela Stojanova
Grigorios Tsoumakas
Karsten Winkler
Michael Witbrock

Nectar Track Programme Committee

Jason Baldridge
Xavier Carreras
Pádraig Cunningham
Marc Deisenroth
Kurt Driessens
Mohammad Ghavamzadeh
Aristides Gionis
David R. Hardoon
Kristian Kersting
Roni Khardon
Christina Leslie
David Page
Liva Ralaivola
Steffen Rendle
Burr Settles
Ivan Titov
Gyorgy Turan
Jean-Philippe Vert
Zhao Xu

Additional Reviewers

Artur Abdullin
Ildefons Magrans de Abril
Claudia d'Amato
Haitham B. Ammar
Fabrizio Angiulli
Josh Attenberg
Mohamad Azar
Luca Baldassarre
Martin Becker
Jaafar Ben-Abdallah
Battista Biggio
Sam Blasiak
Brigitte Boden
Janez Brank
Forrest Briggs
Giulia Bruno
Samuel Rota Bulò
Luca Cagliero
Rui Camacho
Hong Cao
Loic Cerf
Wing Kwan Chan
Anveshi Charuvaka
Silvia Chiusano
Michele Coscia
Dominik Dahlem
Xuan-Hong Dang
Hongbo Deng
Nicola Di Mauro
Luca Didaci
Huyen T. Do
Gauthier Doquire

Nikolaos Engonopoulos
Chaosheng Fan
Kai Fan
Wei Feng
Carlos Ferreira
Raphael Fonteneau
Benoît Frénay
Sergej Fries
David Fuhry
Fabio Fumarola
Victor Gabillon
Esther Galbrun
Shenghua Gao
Aurélien Garivier
Konstantinos Georgatzis
Christos Giatsidis
Tatiana Gossen
Quanquan Gu
Francesco Gullo
Manish Gupta
Basheer Hawwash
Jingrui He
Todd Hester
Xin Huang
Yi Huang
Dino Ienco
Roberto Interdonato
Baptiste Jeudy
Lili Jiang
Xueyan Jiang
Michael Kamp
Emilie Kaufmann
Fabian Keller
Ryan Kiros
Julia Kiseleva
Hannes Korte
Aris Kosmopoulos
Petra Kralj Novak
Philipp Kranen
Hardy Kremer
Anastasia Krithara
Denis Krompass
Aggeliki Lazaridou
Florian Lemmerich
Patrick Lessman
Guy Lever
Lin Liu
Grigorios Loukides
Cécile Low-Kam
Thomas Low
Frangiskos Malliaros
Fernando Martinez-Plumed
Ida Mele
João Mendes Moreira
Glauber Marcius Menezes
Barbora Micenková
Pasquale Minervini
Joseph Modayil
Anna Monreale
Tetsuro Morimura
James Neufeld
Minh Nhut Nguyen
Maximilian Nickel
Inna Novalija
Christopher Oßner
Aline Marins Paes
Joni Pajarinen
Luca Pappalardo
Eric Paquet
Daniel Paurat
Yuanli Pei
Ruggero G. Pensa
Claudia Perlich
Sergios Petridis
Anja Pilz
Gianvito Pio
Cristiano Grijo Pitangui
Marthinus Christoffel du Plessis
Vid Podpečan
Chiara Pulice
Miao Qiao
M. Jose Ramirez-Quintana
Zeehasham Rasheed
Irene Rodriguez Lujan
Bernardino Romera-Paredes
Giulio Rossetti
Natali Ruchansky
Patricia Iglesias Sánchez
Tanwishta Saha
Esin Saka
Antonio Bella Sanjuán
Jan Schlüter
Chun-Wei Seah
Wei Shen
Marcin Skowron
Eleftherios Spyromitros-Xioufis
Tadej Stajner
Daniela Stojanova
Hongyu Su
Yizhou Sun
Akiko Takeda
Frank W. Takes
Jafar Tanha
Claudio Taranto
Sep Thijssen
Stamatina Thomaidou
Daniel Trabold
Mitja Trampus
Roberto Trasarti
Erik Tromp
Yuta Tsuboi
Duygu Ucar
Michal Valko
Ugo Vespier
Silvia Villa
Yana Volkovich
Byron C. Wallace
Jun Wang
Xiaodan Wang
Pascal Welke
Makoto Yamada
Xintian Yang
Jihang Ye
Eric Yeh
Guoxian Yu
Yaoliang Yu
Elias Zavitsanos
Wei Zhang
Tingting Zhao

Table of Contents – Part II

Privacy and Security

Rankings and Recommendations

Reinforcement Learning and Planning

Rule Mining and Subgroup Discovery

Semi-Supervised and Transductive Learning

Sensor Data

Sequence and String Mining

Social Network Mining

Spatial and Geographical Data Mining

Statistical Methods and Evaluation

Time Series and Temporal Data Mining

Transfer Learning

System Demonstrations Track

Nectar Track

Table of Contents – Part I

Invited Talks

Association Rules and Frequent Patterns

Bayesian Learning and Graphical Models

Classification

Dimensionality Reduction, Feature Selection and Extraction

Distance-Based Methods and Kernels

Ensemble Methods

Graph and Tree Mining

Large-Scale, Distributed and Parallel Mining and Learning

Multi-Relational Mining and Learning

Multi-Task Learning

Natural Language Processing

Online Learning and Data Streams

AUDIO: An Integrity *Audi*ting Framework of *O*utlier-Mining-as-a-Service Systems

Ruilin Liu[1], Hui (Wendy) Wang[1], Anna Monreale[2], Dino Pedreschi[2],
Fosca Giannotti[3], and Wenge Guo[4]

[1] Stevens Institute of Technology, NJ, USA
[2] University of Pisa, Pisa, Italy
[3] ISTI-CNR, Pisa, Italy
[4] New Jersey Institute of Technology, NJ, USA
{rliu3,Hui.Wang}@stevens.edu, {annam,pedre}@di.unipi.it,
fosca.giannotti@isti.cnr.it, wenge.guo@njit.edu

Abstract. Spurred by developments such as cloud computing, there has been considerable recent interest in the data-mining-as-a-service paradigm. Users lacking in expertise or computational resources can outsource their data and mining needs to a third-party service provider (server). Outsourcing, however, raises issues about *result integrity*: how can the data owner verify that the mining results returned by the server are correct? In this paper, we present *AUDIO*, an integrity auditing framework for the specific task of distance-based outlier mining outsourcing. It provides efficient and practical verification approaches to check both completeness and correctness of the mining results. The key idea of our approach is to insert a small amount of artificial tuples into the outsourced data; the artificial tuples will produce artificial outliers and non-outliers that do not exist in the original dataset. The server's answer is verified by analyzing the presence of artificial outliers/non-outliers, obtaining a probabilistic guarantee of correctness and completeness of the mining result. Our empirical results show the effectiveness and efficiency of our method.

1 Introduction

Advances in networking technologies have triggered a new computing paradigm called cloud computing. It allows data owners, especially the ones who have large volume of data but limited budget for data analysis, to outsource their data and data mining needs to a third-party service provider. This is referred as the *data-mining-as-a-service* ($DMAS$) model [26,30]. The model allows data owners to leverage hardware and software solutions provided by $DMAS$ providers, without developing their own. There are a few active cloud-based $DMAS$ projects in industry. For example, Google provides cloud-based Google Prediction APIs [2].

Among various types of data mining applications, outlier mining, a classic data mining problem, has seen the possibility to be married with the $DMAS$ paradigm. Outlier mining has been in critical need in many real-world applications such as credit card fraud detection, discovery of criminal activities, weather

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 1–18, 2012.

prediction, marketing and customer segmentation. The task of outlier mining is to find data objects that do not comply with the general patterns of the majority. Sometimes outliers are data errors whose existence may affect the data analysis [8,9]. The problem of outlier detection has been widely studied in the data mining community [3,6,17,27]. It has been shown that detecting outliers is of high computational complexity [17], becoming prohibitive for data of high dimensionality [3]. Although researchers have identified several important optimization techniques [4,27] to improve the efficiency of outlier detection, it is difficult for the data owner who lacks of professional expertise to implement these techniques. $DMAS$ provides a natural solution for such data owner who desires to find outliers from her datasets for analysis purpose.

Although outsourcing is advantageous for the data owner of limited capabilities to achieve sophisticated analysis on their large volume of data, it triggers serious security concerns. One of the major security issues is *result integrity*; the question to be answered is how the data owner (client) of weak computational power can be assured that a service provider (server) returns faithful mining results [1,7]. There are many reasons that the server may return wrong answer. For example, a server may return wrong mining result accidentally due to software bugs, or keep part of the mining result to itself intentionally so that it can sell the retained result to the competitors of the client for profit. There also exists a strong financial incentive for the server to return incorrect answers that require less work and are unlikely to be detected by the client.

We consider two types of service providers, the *semi-honest* server and the *malicious* server, that may return wrong result. The semi-honest server executes the mining algorithm honestly; however, it may modify the outlier mining result by accident. The malicious server executes the mining algorithm unfaithfully (e.g., runs the algorithm on a portion of the dataset) and returns the incorrect result on purpose. Our goal of integrity verification is to enable the client, who is of weak computational power, to verify whether the server that is potentially semi-honest or malicious returns *correct* and *complete* outlier mining result. By *correctness*, we mean that each returned tuple is a true outlier. By *completeness*, we mean that all true outliers are returned by the server.

We design and implement *AUDIO*, a lightweight integrity auditing framework for outlier mining-as-a-service. *AUDIO* includes two entities, the client and the remote, untrusted third-party server. To catch the semi-honest server, before outsourcing, the client constructs a set of artificial tuples that consist of *artificial outliers* (*AO*s) and *artificial non-outliers* (*ANO*s). Both *AO*s and *ANO*s are inserted into the original dataset and sent together to the server. Meanwhile, the client maintains a small piece of auxiliary information locally. After receiving the result from the server, the client verifies the correctness and completeness of the result by analyzing the returned outliers against *AO*s and *ANO*s, and quantifies the probabilistic guarantees of completeness and correctness. There are a few nice properties of our verification techniques. First, incorrect and incomplete answers from the server can be caught with high confidence by a small number of *AO*s and *ANO*s. Second, the complexity of our solution is linear to the number of AOs and $ANOs$, which are independent from the database size. This makes it feasible to efficiently verify the outlier mining results of large databases.

Although it is not new to accomplish verification by inserting counterfeit tuples in other contexts [28,32] and other data mining problems (e.g., association rule mining [31]), it has not been explored of how to verify the result of outsourced outlier mining via counterfeit tuples. Indeed, inserting counterfeit tuples leads to unique challenges to outlier mining. For example, since some outliers in the original dataset may not be outliers anymore in the new dataset after insertion, how to construct counterfeit tuples and design verification techniques to ensure all original outliers are still discoverable in the dataset with newly inserted counterfeit tuples? None of the existing techniques based on insertion of counterfeit tuples can address such challenge.

To our best knowledge, we are the first to address the problem of providing integrity assurance for outsourcing of outlier mining. Our contributions include the following:

(1) To catch the semi-honest server, we propose an artificial-tuple (AT) based approach providing both correctness and completeness guarantee by inserting artificial outliers (AOs) and non-outliers (ANOs) into the original data. We formally quantify both correctness and completeness guarantees, and discuss how to design AOs and ANOs so that the outlierness of AOs and non-outlierness of ANOs do not need to be verified by mining of the dataset.

(2) Inserting AOs and $ANOs$ may change the (non)outlierness of the real tuples. We propose a verification mechanism that will not eliminate any true outlier. We also discuss how to remove the false positive outliers (i.e., the non-outliers returned as outliers) introduced by AOs and $ANOs$ and recover all true outliers efficiently.

(3) We define the possible misbehaviors by the malicious server, and show how the malicious server can defeat the AT-based approach. We also discuss the challenges of designing efficient verification approaches to catch the malicious server.

(4) We complement our analytical results with an extensive set of experiments showing the efficiency and the effectiveness of our AT-based approach.

The paper is organized as following. Sec.2 discusses related work. Sec.3 introduces the preliminaries. Sec.4 presents our AT-based approach to catch the semi-honest server. Sec.5 discusses the limits of our AT-based approach to catch the malicious server, as well as the design of deterministic approaches. Sec.6 presents our experimental results. Sec.7 concludes the paper.

2 Related Work

The issue of integrity assurance for database management was initially raised in the database-as-a-service (DAS) paradigm [14]. The studied problem is how to assure the integrity of SQL query evaluation over the hosted relational databases. The proposed solutions include Merkle hash trees [20,23], signatures on a chain of paired tuples [25], challenge tokens [28], and counterfeit records [32]. [33,34]

extend the security concerns to the data in a metric space and spatial domain. These work share the same integrity concerns in the outsourcing paradigm. However, their focus is different from ours.

The problem of how to protect sensitive data and data mining results in the data-mining-as-a-service (DMAS) paradigm has caught much attention recently. Wong et al. [30] consider utilizing a one-to-n item mapping together with nondeterministic addition of cipher items to protect the identification of individual items in the scenario that frequent pattern mining task is outsourced. Unfortunately, this work has potential privacy flaws; Molloy et al. [22] show how privacy can be breached in the framework of [30]. Tai et al. [29] consider the same scenario and proposed a database transformation scheme that is based on a notion of k-support anonymity. To achieve k-support anonymity, they introduced a pseudo taxonomy tree; the third party server will discover the generalized frequent itemsets instead. Giannotti et al. [12] define a similar privacy model as k-support that requires each item must be indistinguishable from the other $k-1$ items regarding their frequencies. They provide formal privacy analysis of their k-privacy model on both items and frequent patterns. Although these works focus on frequent pattern mining, their encryption techniques can be applied to our work to provide further protection on data and mining results.

Our problem falls into the category of integrity assurance of data-mining-as-a-service (DMAS) paradigm. However, there is very little work in this category. Wong et al. [31] propose auditing techniques for outsourcing of frequent itemset mining. They generate a (small) artificial database such that all itemsets in the database are guaranteed to be frequent and their exact support counts are known. By hosting the artificial database with the original one and checking whether the server has returned all artificial itemsets, the data owner can verify whether the server has returned correct and complete frequent itemsets. To our best knowledge, Wong et al. [31] are the first (and the only) work that addresses the integrity auditing issue of the DMAS paradigm. Their techniques on frequent itemset mining cannot be directly applied to our problem of outlier mining.

3 Preliminaries

Distance-Based Outlier Mining. In this paper, we focus on distance-based outliers, which is well-defined for datasets of any dimension and more suitable for real-world applications where the data distribution does not fit any standard distribution [17]. In particular, an object O in a dataset D is a (p,d)-*outlier* if at least $p\%$ of the objects in D lie greater than distance d from O [17]. Otherwise, O is a non-outlier with regard to the (p,d) setup. For simplicity, we say O is a non-outlier if it is not a (p,d)-*outlier*. We assume $p\% * |D|$ always returns an integer, as it indicates the number of tuples. We use Euclidean distance to measure the distance between two tuples. In particular, given two tuples $t(a_1,\ldots,a_k)$ and $t'(a'_1,\ldots,a'_k)$, $dist(t,t')=\sqrt{\sum_{i=1}^{k}(a_i-a'_i)^2}$.

Outsourcing Setting. In the outlier-mining-as-a-service framework, the data owner (client) outsources her data D, with the configuration of p and d values for (p, d)-outlierness, to the server. The server discovers (p, d)-outliers from D and returns them to the client. We assume the server will return exact outliers instead of approximate ones [16,18].

Types of Dishonest Servers and Assurance Goal. We consider two types of servers that may return invalid answer.

- The *semi-honest* server that runs the outlier mining algorithm on the outsourced dataset faithfully. However, it may return wrong outliers accidentally due to software bugs or human mistakes when collecting the answer.
- The *malicious* server that returns wrong answer intentionally. For example, it may only examine a portion of the outsourced dataset and returns cheaper (and incorrect) answer. Furthermore, the malicious server tries to escape the verification if it knows the details of the verification mechanism.

Let O be the real outliers in the outsourced data D, and O^S be the outliers returned by the server. We define the *precision* of O as $P = \frac{|O \cap O^S|}{|O^S|}$ (i.e., the percentage of returned outliers that are correct), and the *recall* of O as $R = \frac{|O \cap O^S|}{|O|}$ (i.e., the percentage of correct outliers that are returned). Our aim is to catch incorrect answer (i.e., $P < 1$) and incomplete answer (i.e., $R < 1$) with high probability. To this end, we define (α, a)-*completeness* and (β, b)-*correctness.*

Definition 1. *Given a dataset D and a verification method M, let pr_p and pr_r be the probabilities that M catches the server that returns the result of precision $P \leq a$ and recall $R \leq b$ respectively, where $a, b \in [0, 1]$ are given thresholds. We say M can verify (α, a)*-correctness *if $pr_p \geq \alpha$, and can verify (β, b)*-completeness *if $pr_r \geq \beta$, where $\alpha, \beta \in [0, 1]$ are given thresholds.*

4 Artificial Tuple (AT) Based Verification

We develop a verification mechanism using artificial tuples (ATs) to catch the semi-honest server. In particular, before outsourcing the dataset D, the client generates a set of artificial outliers AO and a set of artificial non-outliers ANO respectively. Then the client inserts AO and ANO into the original dataset D, and sends the new dataset $D^+ = D \cup AO \cup ANO$ to the server. Since the semi-honest server cannot distinguish AOs and $ANOs$ from the real tuples in D, it should return all AOs but no $ANOs$, if it is honest. Thus by checking against AOs and ANOs, the client will be able to obtain probabilistic guarantees of completeness and correctness. How to measure the probabilistic guarantees will be discussed in Section 4.2. Next, we first discuss how to construct AOs and ANOs efficiently for preparation of verification (Section 4.1). Second, we discuss how to use AOs and $ANOs$ to accomplish verification (Section 4.2).

4.1 Verification Preparation

Construction of Artificial Non-outliers (*ANO*s). Our *ANO* construction procedure is based on the concept of *close* tuples that we will define soon. Given a tuple t and a distance d, we define two sets, $T_L(t, d)$ that stores all tuples whose distance to t is less than d, and $T_U(t, d)$ that stores all tuples whose distance to t is greater than d. Formally, $T_L(t, d) = \{t'|t' \in D, dist(t, t') < d\}$, and $T_U(t, d) = \{t'|t' \in D, dist(t, t') > d\}$. We say a tuple $t' \in T_L(t, d)$ is the *farthest close neighbor* of tuple t with regard to distance d, if the distance between t and t' is the largest among all tuples in $T_L(t, d)$, and a tuple $t' \in T_U(t, d)$ is the *closest distant neighbor* of tuple t with regard to distance d, if the distance between t and t' is the smallest among all tuples in $T_U(t, d)$. Then we have:

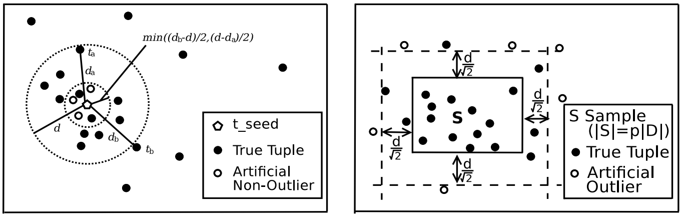
Definition 2. *Given the dataset D of r dimensions and a tuple $t \in D$, let $t_a \in D$ be the farthest close neighbor of t, and $t_b \in D$ be the closest distant neighbor of t. Let P be an r-sphere with t as the centroid, and $min(\frac{d-d_a}{2}, \frac{d_b-d}{2})$ as the radius, where d is the distance parameter of (p, d)-outlier mining, $d_a = dist(t, t_a)$, and $d_b = dist(t, t_b)$. Then we call any tuple $t \in P$ a* close *tuple to t.*

Next, we show that the close tuples of t have the same distance property as t.

Lemma 1. *Given a tuple t and a close tuple t_c of t, for each tuple $t' \neq t, t_c$, it must be true that: (1) if $dist(t, t') < d$, then $dist(t_c, t') < d$; (2) if $dist(t, t') > d$, then $dist(t_c, t') > d$.*

Proof: *Since t_a is the farthest close neighbor of t, for any t' s.t. $dist(t, t') < d$, we have $dist(t, t') \leq d_a < d$, and $dist(t, t_c) < \frac{d-d_a}{2}$, leading to $dist(t', t_c) \leq dist(t, t') + dist(t', t_c) < d_a + \frac{d-d_a}{2} = \frac{d+d_a}{2}$. Since $d_a < d$, then it must be true that $dist(t_c, t') < d$. Similarly, we have $dist(t, t') \geq d_b > d$, and $dist(t, t_c) < \frac{d_b-d}{2}$. Thus, $dist(t_c, t') \geq dist(t, t') - dist(t, t_c) > d_b - \frac{d_b-d}{2} = \frac{d_b+d}{2}$. Since $d_b > d$, then it must be true that $dist(t_c, t') > d$.* □

Based on Lemma 1, next, we prove that any close tuple of tuple t always has the same non-outlierness as t.



(a) ANO construction (b) AO construction

Fig. 1. Construction of *ANO*s and *AO*s

Theorem 1. *Given a dataset D and any tuple $t \in D$ that is a non-(p, d)-outlier, any close tuple of t must be a non-(p, d)-outlier in D.*

The correctness of Theorem 1 is straightforward as t_c and t have the same number of tuples whose distances is greater than the given distance threshold d.

We make use of Theorem 1 to construct ANOs. In particular, we pick a *seed* tuple t_{seed} that is a non-outlier tuple in the original dataset D, and construct its close tuples as ANOs. Fig. 1(a) illustrates the construction procedure of ANOs. To pick t_{seed}, we repeatedly pick a tuple from the dataset randomly, and verify its non-outlierness, until a non-outlier tuple is reached. The probability that t_{seed} will be picked at the x-th trial is $g(x) = (1 - \frac{f_{to}}{n})(\frac{f_{to}}{n})^{x-1}$, where f_{to} and n are the number of outliers and the number of tuples in D. It is straightforward that $1 \leq x \leq n - f_{to}$. Define $\phi = \frac{f_{to}}{n}$. Then $g(x) = (1-\phi)\phi^{x-1}$, where $1 \leq x \leq n - n\phi$. The expected value of x equals to $E(x) = \frac{(n-\phi n)\phi^{n-\phi n+1} - (n-\phi n+1)\phi^{n-\phi n}+1}{(\phi-1)^2}$. As the outliers always take a small portion of the database, ϕ is a small number (e.g., $\phi = 0.05\%$ [24]). Therefore, $E(x) \approx 1$. In other words, it is highly likely that t_{seed} can be picked by the first random trial.

Construction of Artificial Outlier (AOs). Our construction procedure is based on the definition of *distant* tuples. To be more specific, given a r-dimension dataset D and a set of tuples $S \subseteq D$, we say a tuple $t \notin S$ is a *distant* tuple of S if for each tuple t' in S, $dist(t, t') > d$, where d is the parameter setting of (p, d)-outlierness. We have the following lemma to show what kind of tuples can be distant tuples.

Lemma 2. *Given a r-dimension dataset D and a set of tuples $S \subseteq D$, let min_i and max_i be the minimum and maximum value of the i-th $(1 \leq i \leq r)$ attribute of S. Then any tuple $t(a_1, \ldots, a_r) \notin S$ is a* distant *tuple of S if there are k $(1 \leq k \leq r)$ attributes such that on each attribute A_i, $t[A_i] < (min_i - \frac{d}{\sqrt{k}})$ or $a_i > (max_i + \frac{d}{\sqrt{k}})$.*

The correctness of Lemma 2 follows naturally from the distance functions and the properties of the distant tuples. Next, we show that (p, d)-outliers can be generated from the distant tuples.

Theorem 2. *Given the dataset D and a set of tuples $S \subseteq D$, any distant tuple t of S must be a (p, d)-outlier in D if $|S| \geq p|D|$.*

The correctness of Theorem 2 is straightforward as the tuple t of S is of distance of d to p percentage of tuples in D. Based on Theorem 2, we design the AO construction procedure as follows. First, the client picks a sample S of size $[p|D|]$ tuples randomly from D. Second, the client treats S as an r-dimension hypercube $\mathcal{R}$. In $\mathcal{R}$, the edge at the i-th dimension represents the range $[min_i, max_i]$ of the data values in S. Then the client randomly picks k dimensions (possibly $k = 1$) of $\mathcal{R}$. Last, the client expands the k dimensions of $\mathcal{R}$ by $\frac{d}{\sqrt{k}}$ (i.e., change the minimum and maximum value of the i-th attribute to be $min_i - \frac{d}{\sqrt{k}}$ and $max_i + \frac{d}{\sqrt{k}}$). Let the expanded hypercube be $\mathcal{R}'$. Then any tuple t_{ao} that is created outside of $\mathcal{R}'$ must be a (p, d)-outlier of D. Fig. 1(b) illustrates the construction procedure in a 2-dimensional dataset.

Complexity Analysis. *ANO*s can be constructed with at most two passes of the original dataset. The complexity of constructing *AOs* is $O(n)$, where n is the size of D. Therefore, the complexity of the verification preparation is $O(n)$.

4.2 Verification

We realized that (p, d)-outliers in the original dataset D may not be (p, d)-outliers in $D^+ = D \cup ANO \cup AO$, as inserting tuples into the original dataset D may change the (non)outlierness of some true tuples in the original dataset. This may ruin the verification method as the semi-honest server may be wrongly caught as returning incorrect answer. Therefore, the client should ensure that all (p, d)-outliers in D appear in mining of D^+. In this section, we discuss: (1) how the client configures the p parameter so that the true (p, d)-outliers in D are still present in D^+, and (2) how to eliminate the *false positives* (i.e., the tuples returned according to the new p parameter but not (p, d)-outliers in D).

First, we show how to guarantee that the (p, d)-outliers in D are still outliers in D^+ by changing the parameter p of (p, d)-outlier mining.

Theorem 3. *Given a dataset D and a set of AOs and ANOs, any (p, d)-outlier in D must be a (p_1, d)-outlier in $D^+ = D \cup AO \cup ANO$, where $p_1 = \frac{p|D|}{|D^+|}$.*

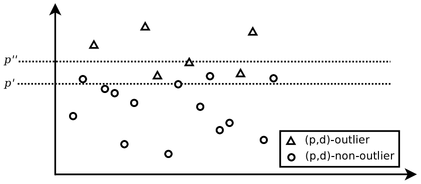


Fig. 2. (p_1, d)-outliers and (p_2, d)-outliers in D^+ VS. (p, d)-outlier in D; O_1: (p_1, d)-outliers in D^+, O_2: (p_2, d)-outliers in D^+.

Proof: *For a true tuple $t \in D$, let m and f be the number of true and artificial tuples (including both AOs and ANOs) whose distance to t is at least d in D^+. Now we prove that it must be true that $\frac{m+f}{|D^+|} \geq p_1 = \frac{p|D|}{|D^+|}$. This can be proven by the following. Since tuple t is a (p, d)-outlier in D, it must be true that $m \geq p|D|$. This naturally leads to that $\frac{m+f}{|D^+|} \geq \frac{p|D|}{|D^+|}$, with $f \geq 0$.* □

Following Theorem 3, the client asks for (p_1, d)-outliers in the outsourced dataset D^+; all outliers in D must appear in the answer if the server is honest. Note that all *AOs* must be appear in (p_1, d)-outliers of D^+ too. However, it is not true that all (p_1, d)-outliers in D^+ must be (p, d)-outliers in D. To eliminate those (p_1, d)-outliers in D^+ that are not (p, d)-outliers in D, we have:

Theorem 4. *Given a dataset D, let $|AO|$ and $|ANO|$ be the numbers of inserted AOs and ANOs respectively. Then any (p_2, d)-outlier in $D^+ = D \cup AO \cup ANO$ must be a (p, d)-outlier in D, where $p_2 = \frac{p|D|+|AO|+|ANO|}{|D^+|}$.*

Proof: *For a tuple $t \in D^+$, let m and f be the number of true and artificial tuples (including both AOs and ANOs) whose distance to t is at least d in D^+. Since the m true tuples must exist in D, we aim to prove that $\frac{m}{|D|} \geq p$. This can be proven as follows. Since t is a (p_2, d)-outlier in D^+, it must hold that $\frac{m+f}{|D^+|} \geq p_2$. This leads to that $m + f \geq p|D| + |AO| + |ANO|$. Since $f \leq |AO| + |ANO|$, it must be true that $m \geq p|D|$. Then the theorem follows.* □

Following Theorem 4, all constructed ANOs must be (p_2, d)-non-outliers in D^+.

Fig. 2 illustrates the relationship among (p_1, d), (p_2, d) outliers in D^+ and (p, d)-outliers in D. For a given tuple t, let per_t be the percentage of tuples in $D^+ = D \cup AO \cup ANO$ whose distance to t is at least d (d: the d parameter for (p, d)-outlierness). Then t will fall into one of the following three categories:

- t is a (p, d)-outlier in D, if $per_t \geq p_2 = \frac{p|D|+|AO|+|ANO|}{|D^+|}$;
- t is a (p, d)-non-outlier in D, if $per_t < p_1 = \frac{p|D|}{|D^+|}$;
- t is either a (p, d)-outlier or a (p, d)-non-outlier in D, otherwise.

Based on both Theorem 3 and Theorem 4, the outsourcing and the verification procedures are designed the following. When outsourcing $D^+ = D \cup AO \cup ANO$ to the server, the client asks for (p_1, d)-outliers and (p_2, d)-outliers, where $p_1 = \frac{p|D|}{|D^+|}$, and $p_2 = \frac{p|D|+|AO|+|ANO|}{|D^+|}$. Note that $O_2 \subseteq O_1$; therefore the client can get both sets of outliers by outsourcing the task once.

After receiving the (p_1, d)-outliers O_1 and (p_2, d)-outliers O_2 from the server, the client verifies the completeness and correctness as the following.

Completeness Verification. To verify whether the server has returned all true outliers, the client checks whether $AO \subseteq O_1$. If $AO \nsubseteq O_1$, the client catches the incomplete outlier answer with 100%; otherwise, the client computes the completeness probability $pr_r = 1 - \alpha^{|AO|}$. To satisfy (α, a)-completeness (i.e., $pr_r \geq a$), the client has to construct AOs of size

$$|AO| = \lceil log_\alpha(1 - a) \rceil. \tag{1}$$

Correctness Verification. For the correctness verification, the client checks whether $ANO \cap O_2$ is empty. If it is not, the client catches the incorrect answer with 100%; otherwise, the client returns the correctness probability $pr_p = 1 - \beta^{|ANO|}$. To meet the (β, b)-correctness (i.e., $pr_p \geq b$) requirement, $|ANO|$ must satisfy that

$$|ANO| = \lceil log_\beta(1 - b) \rceil. \tag{2}$$

Equation 1 and 2 show that $|AO|$ and $ANOs$ are independent of the size of the original dataset; thus our mechanism is especially useful for large datasets.

Furthermore, we observe that it does not need large number of $|ANO|$ and $|AO|$ to catch with high correctness probability the server that changes a small fraction of result. For instance, when $\alpha = 0.99$ (i.e., the server changes 1% of the outliers) and $a = 0.99$ (i.e., the probability to catch such answer is at least 0.99), we only need to add 459 AOs.

Overhead Analysis. The complexity of distinguishing $ANOs$ and AOs from real tuples in the returned result is $O(|ANO| + |AO|)$. Both correctness and completeness verification take $O(1)$ complexity. Therefore, the complexity of verification is $O(|ANO|+|AO|)$. Our empirical study shows that $|ANO|$ and $|AO|$ are relatively small even for large datasets (Section 6), this enables the client to accomplish verification on resource-constrained devices (e.g., smart phones).

4.3 Recovery of True (p, d)-Outliers at Client Side

Since the returned (p_1, d)-outliers may contain some false positive tuples that are not (p, d)-outliers in D, the client will recover the real (p, d)-outliers in D from the returned (p_1, d)-outliers O_1 and (p_2, d)-outliers O_2. To achieve this, first, the client eliminates all AOs (if there is any) from O_2 and O_2 (how to distinguish real tuples, AOs, and $ANOs$ will be discussed in Section 4.4). Let the remaining (p_2, d)-outliers be O'_2. Second, the client examines each tuple in $O_1 - O_2$, and keeps those that are (p, d)-outliers in D. Let these tuples be O_{12}. Then $O_{12} \cup O'_2$ includes all true (p, d)-outliers in D. As will shown in Section 6, the tuples in $O_1 - O_2$ takes a negligible portion of D (less than 0.2%). Therefore, the complexity of outlierness examination of tuples in O_{12} should be $O(|D|)$.

4.4 Auxiliary Data for Verification

Auxiliary Data Sent to the Server. Before the data owner (client) sends out her dataset, she signs each tuple with a cryptographic signature. The signature consists of two sub-signatures: Sig_a and Sig_t. Sig_a provides authenticity guarantee, so that any modification on the original tuples can be caught, while Sig_t is used to distinguish the true tuples from the artificial ones that will be inserted for verification of completeness and correctness. In particular, given a tuple $t(a_1, \ldots, a_n)$, $Sig_a = H(a_1 \oplus \ldots a_n)$, and

$$Sig_t = \begin{cases} H(Sig_a \oplus c_1), \text{ If } t \text{ is a true tuple in } D; \\ H(Sig_a \oplus c_2), \text{ If } t \text{ is an } AO; \\ H(Sig_a \oplus c_3), \text{ If } t \text{ is an } ANO. \end{cases}$$

The client predefines three constants c_1, c_2, and c_3, for the true tuples, the AOs, and the ANOs respectively. The client stores the three constants and the hash function H locally. We require that the hash function H is an efficiently computable collision-resistance hash function. It takes a variable-length input and returns a fixed length binary sequence. Furthermore, it should be difficult to reverse the hash value, i.e., given $H(x)$, it is computational infeasible to compute x. Therefore, the server cannot easily modify the signature.

After completes the computation of signatures, the client sends the dataset to the server. Each tuple in the dataset is associated with its two signatures Sig_a and Sig_t. When the server returns the outlier tuples to the client, he is required to return the two signatures of these tuples too.

Auxiliary Data at the Client Side. The client maintains the hash function H, the number of AOs and $ANOs$, and the three constants c_1, c_2, and c_3 that are used to construct the signatures. It is straightforward that the space overhead of these auxiliary information is negligible. For each outlier tuple t returned by the server, the client computes its signature Sig_a, and the three signatures $Sig_t^1 = H(Sig_a \oplus c_1)$, $Sig_t^2 = H(Sig_a \oplus c_2)$, and $Sig_t^3 = H(Sig_a \oplus c_3)$, by using the three constants c_1, c_2, and c_3 that it has stored locally. Then by comparing the signatures Sig_t^1, Sig_t^2, Sig_t^3 against the Sig_t that is attached to t, the client can distinguish whether t is a true tuples, an AO or an ANO.

5 Discussion

5.1 Dealing with Malicious Server

We consider the following two types of misbehaviors by the malicious server:

(1) **Cheap computation:** The server picks a portion of the dataset $D' \subseteq D$ and return outliers of D' as the outliers of D;
(2) **Verification-aware cheating:** The server is aware of the details of the verification mechanism, and escapes from verification by returning incorrect mining result that fits what the verification mechanism expects.

Unfortunately, our AT-based approach cannot catch the two types of misbehaviors by the malicious server. First, our AT-based approach cannot catch cheap computation. If the dataset portion D' that the server picks satisfies that $D' \subseteq S$, where S is the sample that is used to construct AOs (Section 4.1), the outliers of D' should contain all AOs, i.e., the server can succeed to escape from the completeness verification. Since the probability that $D' \subseteq S$ is $prob = |S|/|D| = p$, where p is the parameter used for (p, d)-outlier mining that is close to 1 in practice [17], the server can escape the completeness verification with very high probability even by mining a small portion of dataset. On the other hand, if the portion D' contains t_{seed} (i.e., the seed tuple that is used to construct ANOs), none of the ANOs will be returned, and thus the server can escape the correctness verification. The probability that the server picks D' that contains t_{seed} is $prob = \frac{|D'|}{|D|}$. The server has to pick a large portion of D if it tries to escape the correctness verification with high probability.

Second, our AT-based approach cannot catch verification-aware cheating. When the server knows the verification mechanism, especially how $AOs/ANOs$ are constructed, it is able to identify $AOs/ANOs$. First, it can find all AOs easily by two passes of the outsourced dataset. In particular, due to the fact that the p value of the (p, d)-outlierness setup is always very close to 1 in practice, the sample S used to construct AOs (Section 4.1) is close to the entire dataset D.

Therefore, the constructed *AOs* will be the skyline points of D. Based on this, the malicious server can easily identify the tuples that have the maximum/minimum values of each attribute as *AOs*. This can be done by traversing the dataset twice (one pass to find maximum/minimum values of each attribute and another one to decide *AOs*). On the other hand, due to the fact that all *ANOs* must be the *close* tuples (Definition 2) to a non-outlier tuple, the malicious server can identify all *ANOs* by searching for non-outlier tuples that are *close* to at least one non-outlier. This requires the server to find all non-outliers, whose complexity is at least as high as the cost of mining all outliers. Though expensive, this enables the server to identify *ANOs* and escape from the correctness verification.

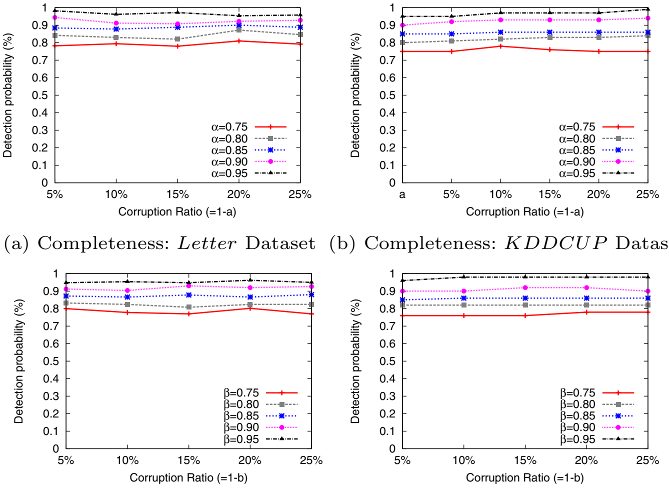
Intuitively, any verification based on transformation of the original dataset (e.g., inserting tuples) may not be able to catch the malicious server as it may launch verification-aware cheating. On the other hand, randomly picking tuples from the original dataset as samples and verifying the outlierness of the samples may resist the verification-based cheating. However, to return high correctness/completeness guarantee, it needs to find a large portion of real outliers, which may not be affordable by the client with limited computational power. We conjecture that to catch a malicious server, especially to catch the verification-aware cheating, the complexity of verification is as expensive as outlier mining.

5.2 From Probabilistic Approach to Deterministic Approach

It is possible that the client may require verification guarantee of 100% certainty. For this case, our AT-based approach cannot meet the requirement, even though it can provide a high probabilistic guarantee. Intuitively, let O^S be the outliers returned by the server, the client can verify the correctness of O^S with 100% certainty by checking whether each tuple in O^S is an outlier in D. The complexity is $O(kn)$, where k is the size of O^S, and n is the size of D. To verify completeness, the client checks whether there exist any tuple in $D - O^S$ that is an outlier. The complexity of completeness verification is $O((n-k)n)$. The total complexity of the deterministic approach is $O(n^2)$, which is as high as outlier mining itself. Although we can optimize the outlier detection procedure [4,27], we have to pay for additional space overhead. We conjecture that the complexity of deterministic approach (in terms of both time and space) is as expensive as outlier mining itself.

6 Experimental Evaluation

We conducted an extensive set of experiments to evaluate both the assurance guarantee and performance of our approach. In particular, we measured: (1) the completeness and correctness guarantee; (2) the verification overhead at the client side, including a) the construction time for AOs and ANOs, b) the verification time to check AOs and ANOs, and c) the time of examining the outlierness of tuples to eliminate false positives; (3) the mining overhead at the server side.



(a) Completeness: *Letter* Dataset (b) Completeness: *KDDCUP* Dataset

(c) Correctness: *Letter* dataset (d) Correctness: *KDDCUP* dataset

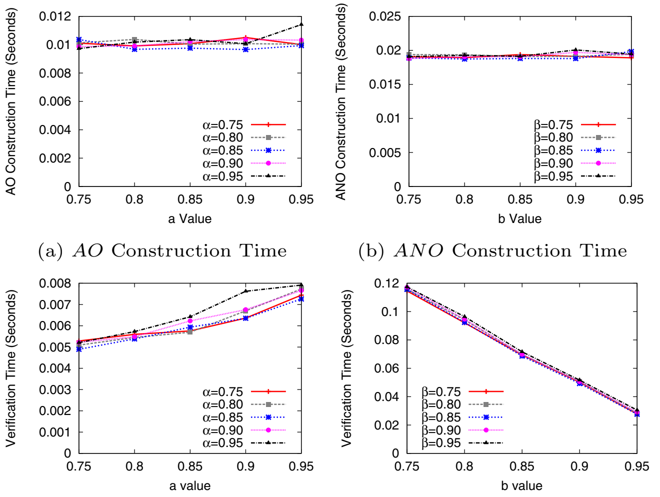
Fig. 3. Robustness of *AT*-based Approach

Setup. In our experiment, we used two datasets, the *Letter* dataset[1] from UCI MLC++ Library that contains $20k$ tuples, and the *KDDCUP* dataset[2] that contains $100k$ tuples. The *Letter* dataset has 16 numerical (integer) attributes. The *KDDCUP* dataset contains 41 (numerical or categorical) attributes. In our experiments, we used five numerical attributes, including `duration`, `dst_bytes`, `flag`, `count`, and `serror_rate` attributes, of the *KDDCUP* dataset. For *Letter* dataset, we used $p = 0.8$ and $d = 15$ that return 1% of the tuples as outliers, while for *KDDCUP* dataset, we used $p = 0.99$ and $d = 5000$ that return 2% of the tuples as outliers. All of our experiments are evaluated on a PC with a 2.4GHz Intel Core 2 Duo CPU and $4GB$ RAM running Windows 7. We implemented the algorithm in Java.

Robustness of the *AT*-Based Approach. We simulate the incomplete result by the semi-honest server as removing 5%, 10%, 15%, 20%, and 25% outliers randomly (i.e., $a = 95\%$, 90%, 85%, 80%, and 75%), and the possible incorrect result as picking 5%, 10%, 15%, 20%, and 25% non-outliers randomly as outliers (i.e., $b = 95\%$, 90%, 85%, 80%, and 75%). Then, we measure the probability of catching these incomplete and incorrect results by our *AT*-based approach by the following: we repeat 500 trials on the *Letter* dataset and 100 trials on the *KDDCUP* dataset. For each trial, first, we insert *AOs* and *ANOs* that are constructed by using our *AT*-based approach. Second, we randomly pick a set of outliers to remove (for producing incomplete answer) and non-outliers to insert (for producing incorrect answer).

[1] `http://www.sgi.com/tech/mlc/db/letter.all`

[2] `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`



(a) *AO* Construction Time (b) *ANO* Construction Time

(c) Verification Time: Completeness (d) Verification Time: Correctness

Fig. 4. *AO*&*ANO* Construction Time and Verification Time, *Letter* dataset

For completeness verification, we examine if all the *AO*s are returns. If at least one *AO* is missing, we count the trial as a *detected* one. Similarly, for correctness verification, we check if the result contains any *ANO*, and count the trial as detected if it does. After we finish all trials, we calculate the detection probability as the $pr_d = \frac{n_{det}}{n_{tr}}$, where n_{det} is the number of detected trials and n_{tr} is the total number of trials.

First, we measured the detection probability of incomplete answer. Figure 3 (a) shows the result on the *Letter* dataset. We observe that the detection probability is always higher than the required α value (i.e., the probability threshold). The same observation also holds for the *KDDCUP* dataset (Figure 3 (b)). We also measured the detection probability of incorrect result. Figure 3 (c) & (d) show the detection probability for *Letter* dataset and *KDDCUP* dataset respectively. It can be easily observed that the detection probability of incorrect result is always better than the required β value, i.e., the correctness probability threshold. This proves the robustness of our *AT*-based approach for both completeness and correctness verification.

Cost Analysis at Client Side. First, we measured the time performance of constructing *AOs* and *ANOs*. Figure 4 (a) shows the *AO* construction time on *Letter* dataset. We observe that the *AO* construction is extremely fast, which needs 0.012 seconds at most even when $\alpha = 0.95$ and $a = 0.95$. Furthermore, when α and a values increase, *AO* construction time increases too, but only slightly. Figure 4 (b) shows the *ANO* construction time on *Letter* dataset. It takes more time than *AO* construction since it needs to find the t_{seed} and verifies whether it is a non-outlier. Nevertheless, it is still fast; it only needs 0.022 seconds at most even when $\beta = 0.95$ and $b = 0.95$. Compared with the mining time (around 200 seconds for the *Letter* dataset), the construction time of *AOs* and *ANOs* is negligible. We also measured the construction time on *KDDCUP* dataset. The construction time of the *KDDCUP* dataset is longer than that of the *Letter* dataset, as the size of the *KDDCUP* dataset is four times larger than that of the *Letter* dataset. However, the *AO*/*ANO* construction is still fast; *AO* construction only needs 0.16 seconds at most, while *ANO* construction only needs 0.035 seconds at most. We omit the result due to limited space.

Second, we measured the verification time at the client side. Figure 4 (c) shows the time of completeness verification on *Letter* dataset. We observed that the time grows when α and a increase. This is straightforward as higher α and a require larger number of *AOs*. Figure 4 (d) shows the time of correctness verification on *Letter* dataset. Contrast to completeness verification, the time of correctness verification decreases with the growth of β and b. This is because with increasing b value, there are fewer real non-outliers inserted as incorrect answer (for simulation). Even though meanwhile larger b value requires more *ANOs*, the number of inserted real non-outliers decreases faster than that of *ANOs*. This leads to the decreasing number of outliers (including real non-outliers and *ANOs*) that the client receives, and consequently less verification time.

Table 1. Number of Tuples Whose Outlierness Are Examined during Post-Processing

a, b	α, β 0.75	α, β 0.8	α, β 0.85	α, β 0.9	α, β 0.95
0.75	1	1	1	2	2
0.80	1	1	2	2	4
0.85	2	2	4	4	5
0.90	4	4	5	5	6
0.95	5	6	6	8	8

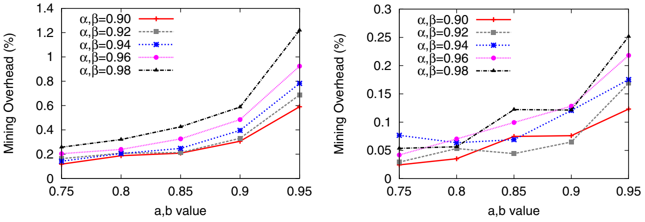
(a) *Letter* Dataset (20K tuples)

a, b	α, β 0.75	α, β 0.8	α, β 0.85	α, β 0.9	α, β 0.95
0.75	2	4	4	6	13
0.80	4	4	6	13	14
0.85	6	8	14	17	23
0.90	14	19	23	23	48
0.95	47	70	77	101	167

(b) *KDDCUP* Dataset (100K tuples)

Third, we measured the performance of outlier recovery at the client side. We first measured the number of tuples whose outlierness needs to be examined in the dataset. Table 1 (a) & (b) show the result of *Letter* dataset and *KDDCUP* dataset respectively. Both tables show that the number of tuples whose outlierness needs to be examined only takes a small portion of the dataset. For example, at most 8 tuples in *Letter* dataset (0.04% of the dataset) and at most 167 tuples in *KDDCUP* dataset (0.16% of dataset) that were examined. Furthermore, we noticed that though it is possible that the tuples that need to be examined can contain true and false positive outliers, in our experiments, all examined tuples are false positive outliers (real non-outliers).

Overhead at Server Side. We measured the mining overhead at the server side as $|T_{D^+} - T_D|/T_D$, where T_D and T_{D^+} are the time of mining outliers from the original database D and the dataset $D^+ = D \cup AO \cup ANO$. Figure 5 (a) and (b) show the result of the *Letter* dataset and *KDDCUP* dataset respectively. We observed that adding *AOs* and *ANOs* does not introduce much mining overhead. For example, it leads to at most additional 1.2% of the original mining time on the *Letter* dataset, and at most additional 0.25% of the original mining time on



(a) *Letter* Dataset (b) *KDDCUP* Dataset

Fig. 5. Mining overhead

the *KDDCUP* dataset. The overhead on *KDDCUP* dataset is much smaller because we insert the same number of artificial tuples for the same α, β, a, b values into a larger dataset. This proves that our *AT*-based approach is more suitable for datasets of large size.

7 Conclusion

Outsourcing mining tasks to a third-party data-mining-as-a-service (DMAS) provider which is potentially untrusted arises serious concern on the integrity of the mining results. In this paper, we focused on verifying the integrity of outlier mining result. We consider two types of server, namely the semi-honest server and the malicious server, that may return incorrect/incomplete outliers. We proposed *AUDIO*, a practical and efficient integrity auditing framework that can provide high correctness and completeness guarantees for the semi-honest server in the DMAS paradigm. We designed efficient approaches of constructing artificial tuples for verification purpose, and demonstrated the efficiency and effectiveness of our approach via an extensive set of experiments.

In the future, we will continue to explore the verification methods for the malicious server. We will investigate whether it is feasible to design efficient verification mechanisms if the server only knows partial details of the verification mechanism, e.g., the server knows there are artificial tuples in the dataset but does not know how these tuples are constructed. We will also examine how to design verification techniques to deal with datasets with updates.

References

1. Cloud Security Alliance. Security Guidance for Critical Areas of Focus in Cloud Computing (2009), `http://www.cloudsecurityalliance.org/guidance/csaguide.pdf`
2. Google Prediction APIs, `http://code.google.com/apis/predict/`
3. Aggarwal, C.C., Yu, P.S.: Outlier detection for high dimensional data. In: SIGMOD (2001)
4. Angiulli, F., Fassetti, F.: Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets. In: TKDD (2009)

5. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of ACM 45 (1998)
6. Barnett, V., Lewis, T.: Outliers in Statistical Data. John Wiley and Sons (1994)
7. Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., Molina, J.: Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control. In: CCSW (2009)
8. Cuzzocrea, A., Wang, W.: Approximate range-Sum query answering on data cubes with probabilistic guarantees. In: JIIS, vol. 27 (2007)
9. Cuzzocrea, A., Wang, W., Matrangolo, U.: Answering Approximate Range Aggregate Queries on OLAP Data Cubes with Probabilistic Guarantees. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 97–107. Springer, Heidelberg (2004)
10. Du, J., Wei, W., Gu, X., Yu, T.: Runtest: assuring integrity of dataflow processing in cloud computing infrastructures. In: ASIACCS (2010)
11. Du, W., Jia, J., Mangal, M., Murugesan, M.: Uncheatable grid computing. In: ICDCS (2004)
12. Giannotti, F., Lakshmanan, L.V., Monreale, A., Pedreschi, D., Wang, H.: Privacy-preserving mining of association rules from outsourced transaction databases. In: SPCC (2010)
13. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal of Computing 18 (1989)
14. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: SIGMOD (2002)
15. Jeevanand, E.S., Nair, N.U.: On determining the number of outliers in exponential and pareto samples. On determining the number of outliers in exponential and Pareto samples. Stat. Pap. 39 (1998)
16. Papadimitriou, S., Kitawaga, H., Gibbons, P.B., Faloutsos, C.: LOCI: Fast Outlier Detection Using the Local Correlation Integral. In: ICDE (2002)
17. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: VLDB (1998)
18. Kollios, G., Gunopulos, D., Koudas, N., Berchtold, S.: An Efficient Approximation Scheme for Data Mining Tasks. In: ICDE (2001)
19. Kreibich, C., Crowcroft, J.: Honeycomb: creating intrusion detection signatures using honeypots. SIGCOMM Computer Communication Review 34 (2004)
20. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD (2006)
21. Liu, K., Giannella, C., Kargupta, H.: An attacker's view of distance preserving maps for privacy preserving data mining. In: PKDD (2006)
22. Molloy, I., Li, N., Li, T.: On the (in)security and (im)practicality of outsourcing precise association rule mining. In: ICDM (2009)
23. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. Trans. Storage 2 (May 2006)
24. Nguyen, H.V., Gopalkrishnan, V., Assent, I.: An Unbiased Distance-Based Outlier Detection Approach for High-Dimensional Data. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 138–152. Springer, Heidelberg (2011)
25. Pang, H., Jain, A., Ramamritham, K., Tan, K.-L.: Verifying completeness of relational query results in data publishing. In: SIGMOD (2005)
26. Qiu, L., Li, Y., Wu, X.: Protecting business intelligence and customer privacy while outsourcing data mining tasks. Knowledge Information System 17(1) (2008)

27. Ramaswamy, S., Rastogi, R., Shim, K., Aitrc: Efficient algorithms for mining outliers from large data sets. In: SIGMOD (2000)
28. Sion, R.: Query execution assurance for outsourced databases. In: VLDB (2005)
29. Tai, C.-H., Yu, P.S., Chen, M.-S.: k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In: SIGKDD (2010)
30. Wong, W.K., Cheung, D.W., Hung, E., Kao, B., Mamoulis, N.: Security in outsourcing of association rule mining. In: VLDB (2007)
31. Wong, W.K., Cheung, D.W., Kao, B., Hung, E., Mamoulis, N.: An audit environment for outsourcing of frequent itemset mining. PVLDB 2 (2009)
32. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: VLDB (2007)
33. Yiu, M.L., Assent, I., Jensen, C.S., Kalnis, P.: Outsourced Similarity Search on Metric Data Assets. TKDE 24 (2012)
34. Yiu, M.L., Ghinita, G., Jensen, C.S., Kalnis, P.: Enabling search services on outsourced private spatial data. VLDB J. 19 (2010)

Differentially Private Projected Histograms: Construction and Use for Prediction

Staal A. Vinterbo[*]

Division of Biomedical Informatics, University of California San Diego
San Diego, CA, USA
sav@ucsd.edu

Abstract. Privacy concerns are among the major barriers to efficient secondary use of information and data on humans. Differential privacy is a relatively recent measure that has received much attention in machine learning as it quantifies individual risk using a strong cryptographically motivated notion of privacy. At the core of differential privacy lies the concept of information dissemination through a randomized process. One way of adding the needed randomness to any process is to pre-randomize the input. This can yield lower quality results than other more specialized approaches, but can be an attractive alternative when *i.* there does not exist a specialized differentially private alternative, or when *ii.* multiple processes applied in parallel can use the same pre-randomized input.

A simple way to do input randomization is to compute perturbed histograms, which essentially are noisy multiset membership functions. Unfortunately, computation of perturbed histograms is only efficient when the data stems from a low-dimensional discrete space. The restriction to discrete spaces can be mitigated by discretization; Lei presented in 2011 an analysis of discretization in the context of M-estimators. Here we address the restriction regarding the dimensionality of the data. In particular we present a differentially private approximation algorithm for selecting features that preserve conditional frequency densities, and use this to project data prior to computing differentially private histograms. The resulting projected histograms can be used as machine learning input and include the necessary randomness for differential privacy. We empirically validate the use of differentially private projected histograms for learning binary and multinomial logistic regression models using four real world data sets.

1 Introduction

Concerns regarding individual privacy are among the major barriers to efficient secondary use of information and data on humans. One of the main difficulties associated with implementation of privacy preserving measures is the quantification of incurred risk to the individual. Differential privacy [5] is a relatively recent measure that has received much attention in the machine learning community

[*] This work was supported by NIH grants R01 LM07273 and U54 HL108460. Thanks go to the anonymous reviewers for their comments.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 19–34, 2012.

as it quantifies individual risk using a strong and cryptographically motivated notion of privacy. For an overview of contributions to the growing number of differentially private methods and theory, we defer to [4,6]. Central to differential privacy is the notion of information dissemination through a randomized process. For a given process that accesses private data and produces information, randomization can be introduced in several places: input perturbation before the data flows into the process, during the computation such as in objective function perturbation [3], and after the process has computed the results but before they are presented [5]. For a given process, analyzing and modifying it so that it yields differential privacy at a given level, and providing a quantification of the inevitable loss in result quality, can be difficult and require sophisticated analysis. These analyses sometimes impose restrictions, as in the type of regularizer in Chaudhuri et al.'s empirical risk minimization [3], and sometimes particular processes cannot be made differentially private in a useful manner, for example computing all k-way marginals [20].

As we will see, achieving a specified level of privacy by perturbing the input to an unconstrained process can yield lower quality results than applying a specialized differentially private version at a fixed privacy level. Nevertheless, this approach to achieving differential privacy can be attractive when *i.* there does not exist a specialized and tailored differentially private alternative, or when *ii.* multiple applications of differentially private processes can be replaced by one or more non-private processes that use a single perturbed instance of the input; the reason is that differential privacy risk for parallel processes composes additively in general. Furthermore, input perturbation can be seen as a noisy information release; having access to information an analysis is based on allows reproduction of results, a very important principle in science and research.

A simple way of producing randomized input from data is to compute a perturbed histogram [5], which essentially is a noisy multiset membership function for the collection of points constituting the data. In order to achieve differential privacy in a histogram by perturbation, noise is added not only to non-zero multiset memberships, but also to originally zero-valued memberships. Consequently, the support of a perturbed histogram extends to virtually all possible elements. This, and the fact that the goal of histogram perturbation includes making originally zero-valued entries indistinguishable from unit-valued entries, perturbed histograms are in practice only applicable to low dimensional and discrete spaces where data is "concentrated", i.e., when there are many duplicates.

The restriction to discrete data can be mitigated by discretization; Lei [15] analyzes discretized histograms in the context of M-estimators and shows consistency results when the discretization bandwidth is chosen carefully. In the following we address the restriction to low dimensionality in the context of learning classifiers. In particular we *a.* present a differentially private approximation algorithm for projecting data onto k features that maximally preserve conditional frequency densities, *b.* show how we can combine projection with thresholded histogram computation to produce histograms that can be used for learning probabilistic classifiers, and *c.* empirically show the viability of building binary and multinomial logistic regression classifiers on differentially private projected histograms. We also show how to compute thresholded histograms in a way that exhibits optimal asymptotic time complexity for data from bounded size spaces.

Other Related Work. Jagadish et al. [13] study binning for histograms in a non-private setting and present optimality results for broad classes of query types. Mohammed et al. [17] and Xiao et al. [25] approach the coarsening of information granularity for differentially private histograms by clustering attribute codomains in a hierarchical manner much like in construction of classification trees. Hay et al. [12] explore tailoring of histograms for answering multiple queries for which we have consistency constraints, for example given as subset sums of query results. Similarly, Xu et al. [26] address the clustering of attribute codomains for a given sequence of counts from a data base, using among others ideas from [13]. The approach presented here is complementary to attribute codomain clustering methods in that the principal method for achieving courser information granularity is instead projection onto a subset of dimensions, consequently not requiring any binning or clustering for categorical attributes. Barak et al. [1] investigate differentially private k-way contingency tables. The projection algorithm in this work can be used as an approximation algorithm for finding a k-way contingency table that reflects a partition of the data with maximum discrimination among elements, and in this sense is related to rough set [18] "reducts". This suggests that projected histograms as computed in this work can also contribute to differentially private rough set theory.

2 Methods

Let $[n] = \{1, 2, \ldots, n\}$, and let $V_i \subseteq U$ for $i \in [d]$ be $d > 0$ finite subsets of some set U. Also let $|V_i| \geq 2$ for all i. We then define $V = V_1 \times V_2 \times \cdots \times V_d$ to be a d-dimensional space of size $m = \prod_{i\in[d]} |V_i|$. A *data set* $\mathcal{D}$ is a collection or multiset of points (records) from V and can be represented by a function $h_{\mathcal{D}} : V \to \mathbb{N}$ counting the number of occurrences of a point in $\mathcal{D}$. A *histogram* is a function $h : V \to \mathbb{R}$, and as such can be seen as a generalization of a multiset. The definition of privacy we use is Differential Privacy [7] and can be stated as follows.

Definition 1. *A randomized algorithm A is ϵ-differentially private if for any measurable set of outputs* $\mathbf{S}$*,*

$$P\left(A(\mathcal{D}) \in \mathbf{S}\right) \leq e^{\epsilon} P\left(A(\mathcal{D}') \in \mathbf{S}\right)$$

where $\mathcal{D}, \mathcal{D}'$ are any two databases of n records that share $n-1$ records in common. The probability is taken over the randomness in A.

The value ϵ is the quantification of individual privacy risk. An important probability distribution in the context of Differential Privacy is the Laplace distribution $\mathrm{Lap}(\mu, \lambda)$ with density

$$f_{\mathrm{Lap}}(r|\mu, \lambda) = \frac{1}{2\lambda} \exp\left(-\frac{|r-\mu|}{\lambda}\right). \tag{1}$$

The mean of this distribution is μ and the variance is $2\lambda^2$. Let $L \sim \mathrm{Lap}(0, 2/\epsilon)$. Then

$$P(L > x) = \begin{cases} \frac{1}{2}\exp\left(-\frac{\epsilon x}{2}\right) & \text{if } x > 0 \\ 1 - \frac{1}{2}\exp\left(\frac{\epsilon x}{2}\right) & \text{otherwise.} \end{cases} \tag{2}$$

In the following we defer proofs to appendix A.

2.1 Differentially Private Histograms

We can create a perturbed version of $h_{\mathcal{D}}$ by adding a perturbation $r(\mathbf{x})$ distributed according to the Laplace distribution, and subsequently threshold by a data-value independent τ to suppress small as well as negative entries. Thresholding by τ has practical implications: the multi-set analogy breaks down with negative entries, learning predictive models from histograms with negative values becomes non-standard, and both the computational effort and space needs associated with thresholded histograms is in practice reduced. Lei [15] proposes $\tau = A \log n/\epsilon$ on the intuition that the maximal noise will be $O(\log n/\epsilon)$. We follow Lei in this definition of τ. Formally,

$$\tilde{h}_{\mathcal{D},\epsilon}(\mathbf{x}) = h_{\mathcal{D}}(\mathbf{x}) + r(\mathbf{x}) \tag{3}$$

$$\tilde{h}_{\tau,\epsilon,\mathcal{D}}(\mathbf{x}) = \begin{cases} \tilde{h}_{\mathcal{D},\epsilon}(\mathbf{x}) & \text{if } \tilde{h}_{\mathcal{D},\epsilon}(\mathbf{x}) > \tau \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

As we do not consider the size of $\mathcal{D}$ private, the subsequent proposition follows directly from results in [5].

Proposition 1. *If τ is only dependent on n and $r(\mathbf{x})$ is distributed according to $\mathsf{Lap}(0, 2/\epsilon)$, then (3) and (4) yield ϵ-differential privacy.*

Since $\tilde{h}_{\tau,\epsilon,\mathcal{D}}$ is constructed in a privacy preserving manner, the data set $\tilde{\mathcal{D}}$ obtained by $h_{\tilde{\mathcal{D}}}(\mathbf{x}) = \lceil \tilde{h}_{\tau,\epsilon,\mathcal{D}}(\mathbf{x}) - 0.5 \rceil$ is privacy preserving.

We have that the value $\tilde{h}_{\tau,\epsilon,\mathcal{D}}(\mathbf{x})$ is distributed according to $\mathrm{Lap}(h_{\mathcal{D}}(\mathbf{x}), 2/\epsilon)$ if $r(\mathbf{x})$ is distributed according to $\mathrm{Lap}(0, 2/\epsilon)$. Consequently, $P(\tilde{h}_{\mathcal{D},\epsilon}(\mathbf{x}) > \tau) = P(L > \tau - h_{\mathcal{D}}(\mathbf{x}))$, where $L \sim \mathrm{Lap}(0, 2/\epsilon)$. Applying (2), the probability of a point $\mathbf{x}$ in V making it into the support S of $\tilde{h}_{\tau,\epsilon,\mathcal{D}}$ is

$$P(\mathbf{x} \in S) = P(L > \tau - h_{\mathcal{D}}(\mathbf{x})) = \begin{cases} \frac{1}{2}\left(\frac{\exp\left(\frac{\epsilon h_{\mathcal{D}}(\mathbf{x})}{2}\right)}{n^{A/2}}\right) & \text{if } \tau > h_{\mathcal{D}}(\mathbf{x}), \\ 1 - \frac{1}{2}\left(\frac{\exp\left(\frac{\epsilon h_{\mathcal{D}}(\mathbf{x})}{2}\right)}{n^{A/2}}\right)^{-1} & \text{otherwise.} \end{cases} \tag{5}$$

If we now let $Y_i \sim \mathrm{Bernoulli}(p_i)$, where $p_i = P(\tilde{h}_{\mathcal{D},\epsilon}(\mathbf{x}_i) > \tau)$ for $\mathbf{x}_i \in V$, we get that the expected size of S is

$$E[|S|] = E[\sum_i Y_i] = \sum_{j=0}^{\infty} |h_{\mathcal{D}}^{-1}(j)| P(L > \tau - j). \tag{6}$$

Let $n' = |\textsc{Supp}(h_{\mathcal{D}})| = \sum_{j=1}^{\infty} |h_{\mathcal{D}}^{-1}(j)|$. If we assume that $\tau = A \log(n)/\epsilon > 1$, then by (5) and (6)

$$n' + \frac{m - n'}{2n^{A/2}} \geq E[|S|] \geq \frac{n' \exp(\frac{\epsilon}{2})}{2n^{A/2}} + \frac{m - n'}{2n^{A/2}} = \frac{\left(\exp(\frac{\epsilon}{2}) - 1\right) n' + m}{2n^{A/2}}. \qquad (7)$$

In (7) we see that $E[|S|]$ grows linearly in m and consequently exponentially in d.

Efficient Construction and Representation. Since V of size m is a product of d finite sets V_i of sizes v_i, we can implement a bijection enc between V and $[m]$ using a multi-base positional number system that maps an element in $O(d)$ time. In our application, the assumption that d and v_i's are small enough so that m can be bounded to fit into a machine word (of typically 64 bits) is mild. Under this assumption, we now show how construct $\tilde{h}_{\tau,\epsilon,\mathcal{D}}$ in expected time that is linear in the size of $\mathcal{D}$ and the output histogram. This is asymptotically optimal since any algorithm computing a histogram from data must input the data and output the histogram. The bounded integer encoding lets us use radix sort to sort a list of integers in linear time, forming the basis of linear time set related operations. Furthermore, given two small constants c_1 and c_2 (≤ 2) and a size z set of keys Z from $[m]$, we can in $O(z)$ time compute a perfect hash function $f_z : Z \to [\lceil c_1 z \rceil]$ that require $O(c_2 z)$ bits of space and has constant evaluation time [2]. Given this result, we can construct a constant access time and $O(z)$ size "hash map" data structure for $Z \subseteq [m]$ in $O(z)$ time. Let $\mathcal{D}'$ be the encoded version of $\mathcal{D}$ created in $O(nd)$ time, and let S' be the set of n' unique elements in $\mathcal{D}'$, which we can extract from $\mathcal{D}'$ in $O(n)$ time using radix sort. Given S', and the hash map data structure we construct the restriction of $\tilde{h}_{\tau,\epsilon,\mathcal{D}'}$ to S' in $O(n)$ time. Naive computation of the remainder of $\tilde{h}_{\tau,\epsilon,\mathcal{D}'}$ would require enumerating $[m] - S'$. We can avoid this explicit enumeration by first determining how many elements from $[m] - S'$ will enter the histogram. This number is the number $\hat{q}$ of successes in $m - n'$ Bernoulli trials with equal success probability $p = P(L > \tau)$. Then we uniformly sample $\hat{q}$ points S'' from $[m] - S'$ without replacement, and assign each a value v in (τ, ∞) chosen with probability proportional to $f_{\mathrm{Lap}}(v|0, 2/\epsilon)$, forming the second part of $\tilde{h}_{\tau,\epsilon,\mathcal{D}'}$. The efficiency of this scheme depends on the efficiency of sampling the distinct points from $[m] - S'$. Using an efficient method [24] for sequential random sampling we sample $\hat{q} + n'$ elements Q without replacement from $[m]$ in expected $O(\hat{q} + n')$ time. We then compute $Q' = Q - S'$ in $O(\max(\hat{q}, n'))$ time using radix sort, and finally sample $\hat{q}$ elements S'' from Q' without replacement in expected $O(\hat{q})$ time using the efficient method. Noting that using hash maps to store the two parts of $\tilde{h}_{\tau,\epsilon,\mathcal{D}'}$, we can merge them in time linear in the sum of their sizes which is $O(n')$ and $O(\hat{q})$, respectively. Consequently, the total time to construct $\tilde{h}_{\tau,\epsilon,\mathcal{D}'}$ is expected $O(nd) + O(\hat{q} + n')$. We can now let $\tilde{h}_{\tau,\epsilon,\mathcal{D}}(x) = \tilde{h}_{\tau,\epsilon,\mathcal{D}'}(\mathrm{enc}(x))$.

Numeric Data. Numeric data is in general too fine-grained to be suitable for direct applications of histogram-based methods, and discretization is required. There is a trade-off between the granularity of the discretization, dictating the preservation of properties of the original data, and suitability for use in the differentially private histogram sampling approach. Lei [15] studies this trade-off in the context of M-estimators, and shows consistency results for differentially private histograms with non-negative counts (i.e., 0-thresholded) when predictors in [0,1] are discretized with bandwidth $\mathrm{bw}(n,d) = \left(\frac{\log(n)}{n}\right)^{1/(d+1)}$. Based on this, we assume that numeric predictors are bounded. This means that they can scaled into [0,1] in a data-independent manner, and subsequently be discretized into $\lceil 1/\mathrm{bw}(n,d-1) - 0.5\rceil$ interval bins. In order to "reverse" the discretization after histogram sampling, we replace each bin identifier with the bin interval midpoint.

2.2 Differentially Private Discernibility

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n points x_i from some set $\mathcal{U}$, and let $A = \{a_1, a_2, \ldots, a_{d-1}\}$ and $f = a_d$ be d functions $a_i : \mathcal{U} \to V_i$. Also, let $=$ be an equivalence relation on V (and V_i). In this formulation, we can say that point $\mathbf{x}_i$ in $\mathcal{D}$ is simply $(a_1(x_i), \ldots, a_{d-1}(x_i), f(x_i))$.

From a classification perspective, we can think of $\mathcal{D}$ as a lookup table: given $A(x) = (a_1(x_i), \ldots, a_{d-1}(x_i))$ look up the conditional relative frequency densities of labels V_d in $\mathcal{D}$ (the distribution of labels associated with elements in $\mathcal{D}$ matching $A(x)$). This approach will fail for a fraction $(1 - \frac{n}{|V|}) \geq (1 - \frac{n}{2^{d-1}})$ of possible points we could want to classify, which becomes a problem if n is much smaller than $|V|$. For fixed n this fraction can be improved by decreasing d, the number of dimensions used. In the following, we present an approach to doing this in a differentially private manner.

We say that a function g *discerns* a pair $(x,y) \in \mathcal{U}^2$ if $g(x) \neq g(y)$, furthermore, for $X \subseteq \mathcal{U}$

$$\begin{aligned} \pi_X(g) &= \{(x,y) \in X^2 | g(x) \neq g(y)\} \\ \pi_X(S) &= \cup_{a \in S} \pi_X(a), \text{ for } S \subseteq A. \end{aligned} \tag{8}$$

For any set $S \subseteq A$ we have that $E_X(S) = X^2 - \pi_X(S)$ is an equivalence relation on X. We denote the equivalence class containing $x \in X$ as $E_X(S)(x)$. We can now express the conditional relative frequency density of $f(x) = v$ for any $x \in \mathcal{U}$ and $v \in V_f$ given X and S as

$$\hat{p}(f(x) = v|S,X) = \begin{cases} \frac{|E_X(S)(x) \cap f^{-1}(v)|}{|E_X(S)(x)|} & \text{if } |E_X(S)(x)| > 0, \\ \frac{|f^{-1}(v) \cap X|}{|X|} & \text{otherwise.} \end{cases}$$

The above uses the unconditional relative frequency density if we have not seen the covariate pattern $S(x)$ before.

Proposition 2. *Let $S \subseteq A$. Then,*

$$\pi_X(S) \cap \pi_X(f) \supseteq \pi_X(A) \cap \pi_X(f) \implies \hat{p}(f(x) = v|S,X) = \hat{p}(f(x) = v|A,X).$$

for any $x \in X$ and any $v \in V_f$.

Proposition 2 characterizes subsets S of A that preserve conditional relative frequency densities. Note that finding a minimum size S such that $\pi_X(S) \cap \pi_X(f) \supseteq \pi_X(A) \cap \pi_X(f)$ is NP-hard, as it essentially is the problem of covering $\pi_X(f)$ by sets $\pi_X(a)$, $a \in S$. Motivated by Proposition 2, we formalize our attribute selection problem as finding $S \subseteq A$ with cardinality k, that maximizes $|\pi_X(S) \cap \pi_X(f)|$. We call this problem k-discernibility, and in the following present an algorithm for it. To start, note that we can partition any $R \subseteq X^2$ into n (possibly empty) sets $R(x_i) = \{y|(x_i, y) \in R\}$. Therefore, $|R| = \sum_i |R(x_i)|$. Let $\pi_X(S)(x) = \{y|(x,y) \in \pi_X(S)\}$. Then we have that $(x,y) \in \pi_X(S) \cap \pi_X(f) \iff y \in \pi_X(S)(x) \cap \pi_X(f)(x)$. Consequently,

$$F(S) = \frac{|\pi_X(S) \cap \pi_X(f)|}{n} = \sum_{i=1}^{n} F_i(S) \tag{9}$$

where

$$F_i(S) = \frac{|\pi_X(S)(x_i) \cap \pi_X(f)(x_i)|}{n}.$$

Lemma 1. *F is submodular and non-decreasing for any constant $n > 0$.*

Now consider the function PrivateKD in Algorithm 1, and let F be defined by (9) with n being the number of elements in the data base. If we had picked a such that it maximized $F(S \cup \{a\}) - F(S)$ at each step, the resulting NonPrivateKD would have been a $(1 - 1/e)$ approximation algorithm for k-discernibility since F is submodular [8]. The formulation of k-discernibility in terms of F and PrivateKD allows us to essentially reuse Theorem 8.2 in Gupta et al. [10] as follows.

Algorithm 1. The differentially private algorithm for the k-discernibility problem.

```
PrivateKD($A, F, k, \epsilon'$)
$S \leftarrow \emptyset$
$A' \leftarrow A$
for $i = 1$ to $k$ do
    Pick $a$ from $A'$ with $P(a) \propto \exp(\epsilon'(F(S \cup \{a\}) - F(S)))$
    $S \leftarrow S \cup \{b\}$
    $A' \leftarrow A' - \{b\}$
end for
return $S$
```

Theorem 1. PrivateKD *is $4k\epsilon'$-differentially private. Furthermore, if $m = d - 1$ then except with probability $O(1/poly(m))$,* PrivateKD *returns a solution not worse than* $(1 - 1/e)\mathrm{OPT} - O(k \log m/\epsilon')$ *where* $\mathrm{OPT} = F(S^*)$ *where S^* is an optimal solution.*

Corollary 1. *If we set* $\epsilon' = \epsilon/(4k)$ *then* PrivateKD *is* ϵ*-differentially private, and has expected solution quality* $(1 - 1/e)\mathrm{OPT} - O(k^2 \log m/\epsilon)$.

We note that $n(F(S \cup \{a\}) - F(S)) = |\pi_X(f) - \pi_X(S)| - |\pi_X(f) - \pi_X(S \cup \{a\})|$, and that $|\pi_X(f) - \pi_X(S)|$ can be computed in terms of the refinement of the partition of X in induced by S by f. We can exploit this to implement the above algorithm to run in $O(k(d-1)n)$ time by at each step keeping track of, and refining, the partition of X induced by the current solution S.

2.3 Projected Histograms

For data $\mathcal{D}$ with target attribute f, and parameters k, ϵ, and $\gamma \in (0, 1]$, we can construct projected histogram as follows. First we spend $\epsilon_p = (1 - \gamma)\epsilon$ of the risk on PrivateKD, finding a $k + 1$ size set of dimensions (including f) to project $\mathcal{D}$ onto. Subsequently we use the remaining risk $\epsilon_h = \gamma\epsilon$ to construct a differentially private histogram from the projected $\mathcal{D}$. We now discuss how to estimate k and γ if they are not given.

Choosing *k*. Ideally, we are looking for a minimal size k set $S \subseteq A$ such that

$$\frac{|\pi_{\mathcal{D}}(S) \cap \pi_{\mathcal{D}}(f)|}{|\pi_{\mathcal{D}}(f) \cap \pi_{\mathcal{D}}(A)|} \geq 1 - \sigma \tag{10}$$

for some $\sigma \geq 0$. The parameter σ can be seen as a noise suppressing parameter [22]. From a viewpoint of the empirical conditional probability, increasing σ yields a step towards the unconditional prior as equivalence classes are collapsed. Given a σ, we can approximate S for a data set $\mathcal{D}$ by using the NonPrivateKD to find a minimum size set S such that $F(S)/F(A) \geq 1 - \sigma$. Call this algoritm for finding a minimum size set of discerning attributes MDA. Given a non-private data set $\mathcal{D}'$ that is sufficiently close in structure to $\mathcal{D}$ we can now compute $k = |\mathrm{MDA}(f, \mathcal{D}', \sigma)|$. This approach allows the use of non-private data sets deemed appropriate, if they exist. Based on our experience with non-private use of projections [22,23], $\sigma \in [0.1, 0.01]$ is appropriate. We choose $\sigma = 0.1$.

If such a public and representative set is not available, we propose using a simulated artificial data set $\mathcal{D}'$ instead. Assuming that the size q of the range of f is not considered private (a reasonable assumption), we construct $\mathcal{D}'$ by sampling n i.i.d. points as follows. For a point $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ the label x_d is uniformly sampled from $[q]$, and each predictor $i < d$ is assumed distributed as $P(x_i = j|x_d) \sim N(0, 1)$, where $N(0, 1)$ is the standard normal distribution. After sampling, all predictors are scaled into [0,1], and discretized using a bandwidth $\mathrm{bw}(n, d - 1)$. We then estimate k as $|\mathrm{MDA}(f, \mathcal{D}', 0.1)|$.

An alternative to the above is to apply the parameter selection method of Chaudhuri et al. [3]. In this approach, we decide on l possibilities of values for k, and partition $\mathcal{D}$ into $l + 1$ parts. Using PrivateKD with privacy ϵ_p, we then compute S_i from $\mathcal{D}_i$ using value k_i, and evaluate each S_i on $\mathcal{D}_{l+1}$, yielding a utility u_i based on $\frac{|\pi_{\mathcal{D}_{l+1}}(S_i) \cap \pi_{\mathcal{D}_{l+1}}(f)|}{|\pi_{\mathcal{D}_{l+1}}(f) \cap \pi_{\mathcal{D}_{l+1}}(A)|}$ and k_i. Finally, we use the exponential mechanism to choose S_i with probability proportional to $\exp(\epsilon_p u_i)$. According to results by Chaudhuri et al. this procedure yields ϵ_p-differential privacy. As each S_i is computed only using a fraction $\frac{1}{l+1}$ of the available data, this approach is only suitable when n/l is large.

Choosing γ. The parameter γ distributes the privacy budget ϵ between the two tasks of projection and histogram sampling. Intuitively, projecting poorly yields bad model instances even when the modeling is done non-privately. On the other hand, a very diffuse histogram, even when it is on a good projection, will make learning good classifiers difficult. From the above, we know that the quality of projection for a fixed k is determined in part by the fraction of pairs l from $\pi_{\mathcal{D}}(f)$ that are not discerned due to privacy. The quality of the histogram sampling is determined by two probabilities: the probability of a point $\mathbf{x} \in \mathcal{D}$ making it into the support $S = \textsc{Supp}(\tilde{h}_{\tau,\epsilon,\mathcal{D}})$ of the sampled histogram, and the probability of a point $\mathbf{y} \notin \mathcal{D}$ making it in as well. However, applying $h_{\mathcal{D}}(\mathbf{y}) = 0$ to (5) we get that $P(\mathbf{y} \in S) = (2n^{A/2})^{-1}$ and consequently does not depend on ϵ. Hence we will concentrate on finding two values: ϵ_h that yield an acceptable $p = P(\mathbf{x} \in S)$, and ϵ_p that yields an acceptable value for l. Once these have been determined we compute γ as

$$\gamma = \frac{\epsilon_h}{\epsilon_h + \epsilon_p}. \tag{11}$$

Determining ϵ_p. We know from Corollary 1 of Theorem 1 that with a high probability, the solution quality our projection algorithm is $\mathrm{OPT}(1-\exp(-1)) + O(k^2 \log(d-1)/\epsilon)$. This quality is on the scale of F, this means that the number of pairs not discerned due to privacy is $O(nk^2 \log(d-1)/\epsilon)$. Note that the upper bound of the number of pairs f can discern given n elements is $\hat{\pi}(q) = \frac{n^2(q-1)}{2q}$. Using a tuning parameter $B > 0$, we use

$$l \leq \frac{nBk^2 \log(d-1)}{\epsilon_p \hat{\pi}(q)}$$

to relate the fraction of pairs l "lost" due to privacy at level ϵ_p. Consequently, we get

$$\epsilon_p \geq \frac{nBk^2 \log(d-1)}{l \hat{\pi}(q)}$$

for a fixed l. Note that l can be seen as having a similar role as the "overfitting" parameter σ in (10). As the algorithm incurs a loss in discerned pairs of at most $\mathrm{OPT}(1-\exp(1)^{-1})$ without privacy, we choose $l = 0.05$, half the value of σ used for selecting k in order to compensate for this to some degree. We also choose $B = 1/2$ in an ad-hoc manner.

Determining ϵ_h. For a point $\mathbf{x} \in \mathcal{D}$, we have that $P(\mathbf{x} \in S)$ is given by (5). If we let $z = h_{\mathcal{D}}(\mathbf{x})$, then if

$$\epsilon_h \geq \max\left(\frac{\log(n)\ A - 2\log\left(\frac{1}{2\,p}\right)}{z}, \frac{\log(n)\ A - 2\log(2 - 2\,p)}{z}\right)$$

we get $P(\mathbf{x} \in S) \geq p$. What remains is to determine which value z to use. In the absence of other information, we estimate a value for z as follows. Recall that the target is labelled with q labels. We assume that these are uniformly distributed, meaning that the expected number of points in $\mathcal{D}$ that are labelled with the same label is n/q. Furthermore, the projection algorithm aims at approximating the partition induced by the target f, ideally such that all points within an equivalence class induced by the predictors are a subset of an equivalence class induced by f. Also, we assume that predictors are independent given the target. This means that if $p_{\mathbf{x}}$ denotes the probability that a randomly and independently chosen point falls into the equivalence class of $\mathbf{x}$, we can estimate the size of the equivalence class containing $\mathbf{x}$ as $z_{\mathbf{x}} = p_{\mathbf{x}}n/q$. Finally, we assume that each of the $d-1$ predictors is produced from discretizing truncated normally distributed i.i.d. random variables into $1/b$ equally spaced bins after scaling the values to fall into [0,1]. As above, we compute the discretization bandwidth as $b = \mathrm{bw}(n, d-1)$, yielding $s = \lceil 1/b - 0.5 \rceil$ bins. Let p_i be the probability mass of bin i. Then, $p_{\mathbf{x}} = (\sum_{i=1}^{s} p_i^2)^k$ for k predictors. The p_i can be estimated by sampling N points from $N(0,1)$, scaling these into [0,1], partition into s bins c_i, and letting $\hat{p}_i = \frac{|c_i|}{N}$.

3 Experiments

As stated in Section 2.1, given a privacy preserving histogram $\tilde{h}_{\mathcal{T},\epsilon,\mathcal{D}}$, we can generate a privacy preserving data set $\tilde{\mathcal{D}}$ by letting $h_{\tilde{\mathcal{D}}}(\mathbf{x}) = \lceil \tilde{h}_{\mathcal{T},\epsilon,\mathcal{D}}(\mathbf{x}) - 0.5 \rceil$. We performed experiments to elicit the utility of projected histograms as a non-classifier specific platform to build differentially private classifiers on. We did this by comparing the performance of binary and multinomial logistic regression classifiers build on $\tilde{\mathcal{D}}$ versus built on the original $\mathcal{D}$, as well with classifiers based on a particular method for differentially private logistic regression trained on $\mathcal{D}$. We used the R environment environment for statistical computing [19] for all our experiments.

3.1 Setup

Classifiers. The classification tasks in our experiments are both binary and multinomial. For both we used three different classifier learning algorithms. In particular, for binary classification these are OLR – non-private logistic regression using the generalized linear modeling `glm` function of the `stats` [19] library, HLR – an OLR classifier trained on data created from a differentially private projected histogram, and PLR – logistic regression using the differentially private empirical risk minimization [3]. For multinomial classification these are OMLR – non-private multinomial logistic regression using the `multinom` function of the `nnet` [21] package, HMLR – an OMLR classifier trained on data created from a differentially private projected histogram, and PMLR – a classifier constructed from learning an ϵ/q-differentially private PLR classifier for each of the q classes and applying them in parallel to obtain normalized probabilities for each class.

Data Sets. In the experiments we used four data sets. Two intended for binary prediction and two for multi-class prediction. For each of binary and multi-class, one data set is smaller and one is larger.

Iris – a 3 class data set of 4 predictor attributes on 150 cases. The data set describes 4 measurements of three different types of iris plants. The data set is available from the UCI Machine Learning Repository [9] as the "Iris Data Set".

Satellite – a 6-class data set of 36 predictor attributes on 6435 cases. The data set describes a 3x3 neighborhood of pixels in a Landsat satellite multi-spectrum digital image. The six classes are soil types. The data set is available from the UCI Machine Learning Repository [9] as the "Statlog (Landsat Satellite) Data Set".

Infarct – a 2 class data set of 44 predictor attributes on 1753 cases. The data set describes 44 measurements taken from patients presenting with chest pain at the emergency rooms of two hospitals, one in Sheffield, England, and one in Edinburgh, Scotland. The classification is according to whether the patients received a diagnosis of myocardial infarction or not. The data set is not publicly available, and has been used in [14].

Adult – a 2 class data set of 14 predictor variables on 45083 cases. The data set describes measurements on cases taken from the 1994 Census data base. The classification is whether or not a person has an annual income exceeding 50000 USD. The data set is available from the UCI Machine Learning Repository [9] as the "Adult Data Set".

Estimation of Performance. In order to be able to relate performances across binary and multinomial classification tasks, we assessed binary classification quality using the area under the receiver operating curve (AUC), and the generalization of the AUC given by Hand et al. [11] for the multinomial tasks.

Construction of a projected histogram has parameters ϵ – the differential privacy level, k – the number of sub-dimensions to project onto, and γ – which decides the distribution of ϵ among projection and histogram creation. Given ϵ, and the size of the input data, the algorithm can estimate the two latter parameters k and γ. We ran separate ten-fold cross-validation experiments stratified on class labels to *i.* compare classifier performances given a baseline privacy requirement $\epsilon = 1$ and using estimated k and γ, and *ii.* investigate the quality of the estimation applied to produce k and γ, as well as behavior of histogram based classifiers under varying privacy requirements ϵ. Each parameter variation was investigated in independent experiments where the other parameters were set to either to baseline ($\epsilon = 1$) or left for the algorithm to estimate (k and γ). The tuning parameters A and B were left fixed at $A = 1/2$ and $B = 1/2$.

3.2 Results

Figure 1 shows box-and-whisker plots of the cross-validation AUCs for the classifiers on all four data sets (upper row: multi-class, lower row: binary class, left

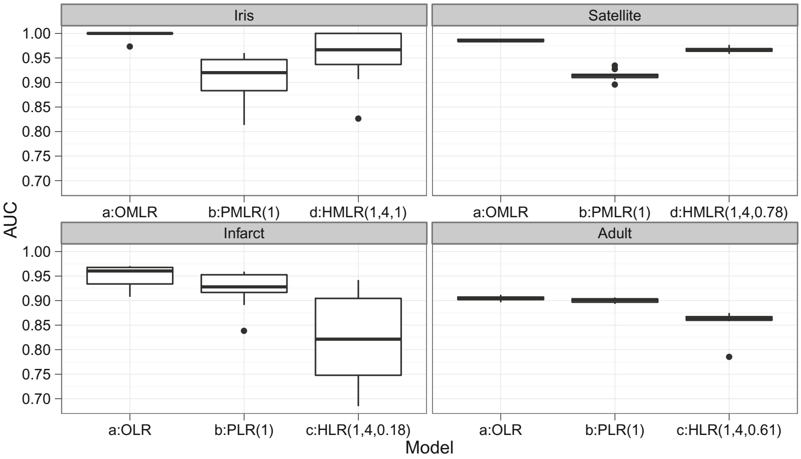


Fig. 1. Box-and-whisker plots of the 10-fold cross-validation AUC performance values for the classifiers on the four data sets. The quantiles in the plots are the standard 25%, 50%, and 75%. The whiskers denote the most extreme points within 1.5 times the respective interquartile ranges (i.e., the 1.5 "rule"), while outliers are denoted by dots.

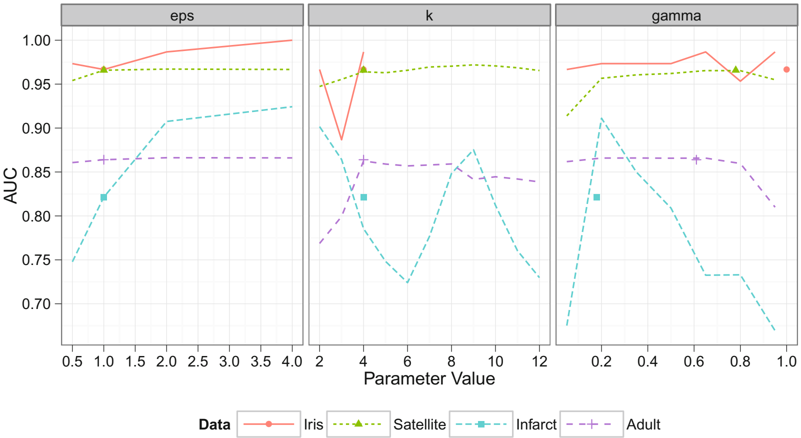


Fig. 2. The median cross-validation AUC for the histogram based method as we vary parameters ϵ, k, and γ on each of the four data sets. The dots represent the baseline/estimated value for the parameter and the AUC value taken from the ϵ series of cross-validation experiments.

column: smaller, right column: larger). Figure 2 shows the behavior of the median cross-validation AUC for the projected histogram method H(M)LR as we vary parameters ϵ, k, and γ.

4 Discussion

We presented a differentially private feature selection algorithm and used it to project data before computing differentially private histograms. We provided a method for distributing the allowed privacy risk between the two tasks.

We experimented with logistic regression in order to investigate the classifier performance loss between using perturbed projected histograms for input perturbation over a specialized differentially private method. While loss is present, as can be seen in the bottom row of Figure 1, projected perturbed histograms offer utility as the performance still is arguably good.

To demonstrate the use of projected histograms when there are no specialized alternatives available, we used projected histograms to construct differentially private multinomial logistic regression models. We know of no differentially private and available algorithm for learning non-binary probabilistic classifiers; as the alternative for q classes, we used estimates from q differentially private binary logistic regression models (we also experimented with using parameters from $q - 1$ models in a softmax function, but this is sensitive to base class selection and performed worse than using q binary model estimates directly). The models learned from perturbed projected histograms outperformed the private multi-model alternative as seen in the top row in Figure 1. This supports that considering using perturbed projected histograms for input perturbation over using multiple applications of specialized methods that each incur separate and additive risk is warranted.

A limitation of our approach is that the parameter k must be estimated. In the middle panel of Figure 2 we see that while in general $k = 4$ yields good performance, it is not optimal. In particular, for the smaller binary class data set, In fact, $k = 2$ would have increased the competitiveness of the binary logistic regression based on histograms significantly. However, the right panel in Figure 2 shows that for the estimated k, the distribution of the allowed privacy risk between projection and histogram computation is near optimal.

For both binary and multinomial regression we chose two data sets, one smaller and one larger. The leftmost panel in Figure 2 shows the expected improvement of histogram based models with increase in allowed privacy risk. This improvement is much stronger for smaller data. Our intuition for this is that for larger data sets, the reduction in performance due to privacy is dominated by the privacy independent loss in performance due to projection and discretization. We speculate that projected histograms might be even better suited for classification tasks in discrete spaces.

As the size of differentially private histograms grows exponentially in the dimensionality of the feature space, efficient representation and computation is of importance. For spaces of size bounded by 2^B where B is the machine word length, we presented an efficient approach to both computation as well as representation of these histograms.

References

1. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In: PODS. pp. 273–282 (2007)
2. Belazzougui, D., Botelho, F., Dietzfelbinger, M.: Hash, displace, and compress. In: Fiat, A., Sanders, P. (eds.) Algorithms - ESA 2009, Lecture Notes in Computer Science, vol. 5757, pp. 682–693. Springer Berlin / Heidelberg (2009)
3. Chaudhuri, K., Monteleoni, C., Sarwate, A.: Differentially private empirical risk minimization. JMLR 12, 1069–1109 (March 2011)
4. Dwork, C.: Differential privacy: A survey of results. Theory and Applications of Models of Computation pp. 1–19 (2008)
5. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: 3rd IACR TCC. pp. 486–503 (2006)
6. Dwork, C., Smith, A.: Differential privacy for statistics : What we know and what we want to learn. J. Privacy and Confidentiality 1(2), 135–154 (2008)
7. Dwork, C.: Differential privacy. In: M. Bugliesi, B. Preneel, V.S., Wegener, I. (eds.) ICALP (2). Lecture notes in computer science, vol. 4052, pp. 1–12. Springer Verlag (2006)
8. Fisher, M., Nemhauser, G., Wolsey, L.: An analysis of approximations for maximizing submodular set functions—ii. Polyhedral combinatorics pp. 73–87 (1978)
9. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
10. Gupta, A., Ligett, K., McSherry, F., Roth, A., Talwar, K.: Differentially private combinatorial optimization. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1106–1125. Society for Industrial and Applied Mathematics (2010)
11. Hand, D.J., Till, R.J.: A simple generalisation of the area under the roc curve for multiple class classification problems. Machine Learning 45, 171–186 (2001), 10.1023/A:1010920819831
12. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histograms through consistency. Proceedings of the VLDB Endowment 3(1-2), 1021–1032 (2010)
13. Jagadish, H., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K., Suel, T.: Optimal histograms with quality guarantees. In: Proceedings of the International Conference on Very Large Data Bases. pp. 275–286. INSTITUTE OF ELECTRICAL & ELECTRONICS ENGINEERS (1998)
14. Kennedy, R.L., Burton, A.M., Fraser, H.S., McStay, L.N., Harrison, R.F.: Early diagnosis of acute myocardial infarction using clinical and electrocardiographic data at presentation: Derivation and evaluation of logistic regression models. European Heart Journal 17, 1181–1191 (1996)
15. Lei, J.: Differentially private m-estimators. In: NIPS. pp. 361–369 (2011)
16. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: FOCS. pp. 94–103 (2007)
17. Mohammed, N., Chen, R., Fung, B., Yu, P.: Differentially private data release for data mining. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 493–501. ACM (2011)

18. Pawlak, Z.: Rough Sets, Theoretical Aspects of Reasoning about Data, Series D: System Theory, Knowledge Engineering and Problem Solving, vol. 9. Kluwer Academic Publishers (1991)
19. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2011), ISBN 3-900051-07-0
20. Ullman, J., Vadhan, S.: Pcps and the hardness of generating synthetic data. In: ECCC. vol. 17, p. 17 (2010)
21. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S. Springer, New York, fourth edn. (2002), iSBN 0-387-95457-0
22. Vinterbo, S., Øhrn, A.: Minimal approximate hitting sets and rule templates. International Journal of Approximate Reasoning 25(2), 123–143 (2000)
23. Vinterbo, S.A., Kim, E.Y., Ohno-Machado, L.: Small, fuzzy and interpretable gene expression based classifiers. Bioinformatics 21(9), 1964–1970 (Jan 2005)
24. Vitter, J.S.: An efficient algorithm for sequential random sampling. ACM Trans. Math. Softw. 13(1), 58–67 (Mar 1987)
25. Xiao, Y., Xiong, L., Yuan, C.: Differentially private data release through multidimensional partitioning. Secure Data Management pp. 150–168 (2011)
26. Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G.: Differentially private histogram publication. In: Proceedings of the IEEE International Conference on Data Engineering (2012)

A Proofs

Proof of Proposition 2. For readability, let $E_B(x) = E_X(B)(x)$ for all $B \subseteq A$, and let $E_f(x) = E_X(f)(x)$. First, we note that $E_A(x) \subseteq E_S(x)$. If $E_S(x) \subseteq E_f(x)$ for any $S \subseteq A$ we have that the proposition holds as

$$E_S(x) \cap f^{-1}(v) = \begin{cases} E_S(x) & \text{if } f(x) = v \text{, and} \\ \emptyset & \text{otherwise.} \end{cases}$$

Hence, in order to prove the proposition we need to show that for $S \subseteq A$

$$(\pi(S) \cap \pi(f) \supseteq \pi(A) \cap \pi(f) \wedge E_S(x) \not\subseteq E_f(x)) \implies E_S(x) = E_A(x). \tag{12}$$

We start by noting that for all $x \in X$ $\pi(S) \cap \pi(f) \supseteq \pi(A) \cap \pi(f) \implies E_s(x) \cup E_f(x) \subseteq E_A(x) \cup E_f(x)$, which in turn implies that $(E_S(x) - E_f(x)) \subseteq E_A(x)$. This means that as $E_S(x) \not\subseteq E_f(x)$ we can pick $y \in E_S(x) \cap E_A(x)$ such that $y \notin E_f(x)$. Assuming the negation of (12) (giving $E_S(x) \neq E_A(x)$) we can pick an additional point $z \in E_S(x) \cap E_f(x)$ such that $z \notin E_A(x)$. This means $(y, z) \notin \pi(S)$ and $(y, z) \in \pi(A) \cap \pi(f)$ which creates a contradiction, from which we conclude that (12) must hold. □

Proof of Lemma 1. (Sketch:) The requirement for F being submodular is that $F(S \cup S') + F(S \cap S') \leq F(S) + F(S')$ for any $S, S' \subseteq A$. For π defined by (8), we have that $A \subseteq B \implies \pi(A) \subseteq \pi(B)$, and $\pi(A \cup B) = \pi(A) \cup \pi(B))$. We then have that $\pi(A) \cap \pi(B) = (\pi(A - B) \cup \pi(A \cap B)) \cap (\pi(B - A) \cup \pi(B \cap A)) \supseteq \pi(A \cap B)$. Consequently, $m(S) = |\pi(S)|$ is submodular and non-decreasing, as is $F(S) = |\pi(S) \cap \pi(f)|/n$ for fixed f and n. □

Proof of Theorem 1. Consider any $X' \subseteq U$ such that $(X' \cup X) - (X' \cap X) = \{\mathbf{x}_i\}$. This means that X' and X only differ in the element $\mathbf{x}_i$. Changing $\mathbf{x}_i$ leads to change in $F_i^X(S) = |\pi(S)(\mathbf{x}_i) \cap \pi(f)(\mathbf{x}_i)|/n$ at most 1, while in $F_j^X(S)$ at most $1/(n-1)$ for $j \neq i$. This means that $F^X = \sum_i F_i^X$ yields $\Delta(F) \leq 2$. The probability $p(a)$ of selecting a from $A - S$ is

$$p(a) = \frac{e^{\epsilon'(F(S \cup \{a\}) - F(S))}}{\sum_{a' \in A-S} e^{\epsilon'(F(S \cup \{a'\}) - F(S))}} = \frac{e^{\epsilon' F(S \cup \{a\})}}{\sum_{a' \in A-S} e^{\epsilon' F(S \cup \{a'\})}} \tag{13}$$

As in the standard argument for privacy of the exponential mechanism, we note that a change of $\Delta(F)$ contributes at most $e^{\epsilon'\Delta(F)}$ to the numerator of (13), and at least $e^{-\epsilon'\Delta(F)}$ to the denominator, yielding $2\Delta(F)\epsilon'$-differential privacy. We are selecting k times, each with $4\epsilon'$-differential privacy, hence $4k\epsilon'$ total differential privacy.

Similarly to McSherry and Talwar's analysis of the exponential mechanism [16], let $s(a) = F(S \cup \{a\}) - F(S)$ for a fixed $S \subseteq A$, let $\mathrm{OPT} = \max_a s(a)$, and $S_t = \{a | \exp(\epsilon s(a)) > t\}$. Then we have that

$$\begin{aligned} P(\overline{S}_{(t/\epsilon')}) &\leq \frac{P(\overline{S}_{(\mathrm{OPT}-t/\epsilon')})}{P(S_{\mathrm{OPT}})} = \frac{\sum_{a \in \overline{S}_{(\mathrm{OPT}-t/\epsilon')}} \exp(\epsilon' s(a))}{\sum_{a \in S_{\mathrm{OPT}}} \exp(\epsilon' s(a))} \\ &\leq \frac{\sum_{a \in \overline{S}_{(\mathrm{OPT}-t/\epsilon')}} \exp(\epsilon'(\mathrm{OPT} - t/\epsilon'))}{\sum_{a \in S_{\mathrm{OPT}}} \exp(\epsilon' \mathrm{OPT})} = \frac{|\overline{S}_{(\mathrm{OPT}-t/\epsilon')}|}{|S_{\mathrm{OPT}}|} \exp(-t) \\ &\leq |A - S| \exp(-t) \leq m \exp(-t). \end{aligned} \tag{14}$$

If we let $t = \log(m) + t'$, we get that $P(\overline{S}_{(\mathrm{OPT}-(\log(m)+t')/\epsilon')}) \leq \exp(-t')$. If we now let $t' = a\log(m)$, we get that $P(\overline{S}_{(\mathrm{OPT}-((a+1)\log(m))/\epsilon')}) \leq \exp(-a\log(m)) = 1/m^a$, and $P(S_{(\mathrm{OPT}-((a+1)\log(m))/\epsilon')}) \geq 1 - 1/m^a$. This means that the probability that the algorithm chooses k times within $S_{(\mathrm{OPT}-((a+1)\log(m))/\epsilon')}$ is at least $1 - k/m^a$, i.e., we choose well except with a probability $O(k/m^a) = O(1/poly(m))$. When we choose well, since F is sub-modular, every choice adds at most a factor $(1 - 1/k)\mathrm{OPT}$ to the final solution quality gap while adding at most $(a+1)\log(m)/\epsilon'$ to the same. Consequently the gap is $\mathrm{OPT}(1 - 1/e) + O(k\log(m)/\epsilon')$. □

Fairness-Aware Classifier with Prejudice Remover Regularizer

Toshihiro Kamishima[1], Shotaro Akaho[1], Hideki Asoh[1], and Jun Sakuma[2]

[1] National Institute of Advanced Industrial Science and Technology (AIST),
AIST Tsukuba Central 2, Umezono 1-1-1, Tsukuba, Ibaraki, 305-8568 Japan
mail@kamishima.net, {s.akaho,h.asoh}@aist.go.jp
http://www.kamishima.net

[2] University of Tsukuba, 1-1-1 Tennodai, Tsukuba, 305-8577 Japan;
and, Japan Science and Technology Agency, 4-1-8, Honcho, Kawaguchi, Saitama,
332-0012 Japan
jun@cs.tsukuba.ac.jp

Abstract. With the spread of data mining technologies and the accumulation of social data, such technologies and data are being used for determinations that seriously affect individuals' lives. For example, credit scoring is frequently determined based on the records of past credit data together with statistical prediction techniques. Needless to say, such determinations must be nondiscriminatory and fair in sensitive features, such as race, gender, religion, and so on. Several researchers have recently begun to attempt the development of analysis techniques that are aware of social fairness or discrimination. They have shown that simply avoiding the use of sensitive features is insufficient for eliminating biases in determinations, due to the indirect influence of sensitive information. In this paper, we first discuss three causes of unfairness in machine learning. We then propose a regularization approach that is applicable to any prediction algorithm with probabilistic discriminative models. We further apply this approach to logistic regression and empirically show its effectiveness and efficiency.

Keywords: fairness, discrimination, logistic regression, classification, social responsibility, information theory.

1 Introduction

Data mining techniques are being increasingly used for serious determinations such as credit, insurance rates, employment applications, and so on. For example, credit scoring is frequently determined based on the records of past credit data together with statistical prediction techniques. Needless to say, such serious determinations must guarantee fairness in both social and legal viewpoints; that is, they must be unbiased and nondiscriminatory in relation to sensitive features such as gender, religion, race, ethnicity, handicaps, political convictions, and so on. Thus, sensitive features must be carefully treated in the processes and algorithms for data mining.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 35–50, 2012.

There are reasons other than the need to avoid discrimination for prohibiting the use of certain kinds of features. Pariser pointed out a problem that friend candidates recommended to him in Facebook were biased in terms of their political convictions without his permission [15]. For this problem, it would be helpful to make recommendations that are neutral in terms of the user's specified feature, i.e., the candidate friends' political convictions. Further, there are features that cannot legally be exploited due to various regulations or contracts. For example, exploiting insider information and customer data are respectively restricted by stock trading regulation and privacy policies.

Several researchers have recently begun to attempt the development of analytic techniques that are aware of social fairness or discrimination [17,3]. They have shown that the simple elimination of sensitive features from calculations is insufficient for avoiding inappropriate determination processes, due to the indirect influence of sensitive information. For example, when determining credit scoring, the feature of race is not used. However, if people of a specific race live in a specific area and address is used as a feature for training a prediction model, the trained model might make unfair determinations even though the race feature is not explicitly used. Such a phenomenon is called a red-lining effect [3] or indirect discrimination [17].

In this paper, we formulate causes of unfairness in data mining, develop widely applicable and efficient techniques to enhance fairness, and evaluate the effectiveness and efficiency of our techniques. First, we consider unfairness in terms of its causes more deeply. We describe three types of cause: *prejudice*, *underestimation*, and *negative legacy*. Prejudice involves a statistical dependence between sensitive features and other information; underestimation is the state in which a classifier has not yet converged; and negative legacy refers to the problems of unfair sampling or labeling in the training data. We also propose measures to quantify the degrees of these causes.

Second, we then focus on indirect prejudice and develop a technique to reduce it. This technique is implemented as regularizers that restrict the learner's behaviors. This approach can be applied to any prediction algorithm with discriminative probabilistic models, such as logistic regression. In solving classification problems that pay attention to sensitive information, we have to consider the trade-off between the classification accuracy and the degree of resultant fairness. Our method provides a way to control this trade-off by adjusting the regularization parameter. We propose a *prejudice remover regularizer*, which enforces a determination's independence from sensitive information.

Finally, we perform experiments to test the effectiveness and efficiency of our methods. We evaluate the effectiveness of our approach and examine the balance between prediction accuracy and fairness. We demonstrate that our method can learn a classification model by taking into account the difference in influence of different features on sensitive information.

Note that in the previous work, a learning algorithm that is aware of social discrimination is called *discrimination-aware mining*. However, we hereafter use the terms, "unfairness" / "unfair" instead of "discrimination" / "discriminatory" for two reasons. First, as described above, these technologies can be used for various purposes other than avoiding discrimination. Second, because the term *discrimination* is frequently used for the meaning of classification in the data mining literature, using this term becomes highly confusing.

We discuss causes of unfairness in section 2 and propose our methods for enhancing fairness in section 3. Our methods are empirically compared with a 2-naïve-Bayes method proposed by Calders and Verwer in section 4. Section 5 shows related work, and section 6 summarizes our conclusions.

2 Fairness in Data Analysis

After introducing an example of the difficulty in fairness-aware learning, we show three causes of unfairness and quantitative measures.

2.1 Illustration of the Difficulties in Fairness-Aware Learning

We here introduce an example from the literature to show the difficulties in fairness-aware learning [3], which is a simple analytical result for the data set described in section 4.2. The researchers performed a classification problem. The sensitive feature, S, was gender, which took a value, Male or Female, and the target class, Y, indicated whether his/her income is High or Low. There were some other non-sensitive features, X. The ratio of Female records comprised about 1/3 of the data set; that is, the number of Female records was much smaller than that of Male records. Additionally, while about 30% of Male records were classified into the High class, only 11% of Female records were. Therefore, Female–High records were the minority in this data set.

In this data set, we describe how Female records tend to be classified into the Low class unfairly. Calders and Verwer defined a *discrimination score* (hereafter referred to as the Calders-Verwer score (CV score) by subtracting the conditional probability of the positive class given a sensitive value from that given a non-sensitive value. In this example, a CV score is defined as

$$\Pr[Y{=}\mathsf{High}|S{=}\mathsf{Male}] - \Pr[Y{=}\mathsf{High}|S{=}\mathsf{Female}].$$

The CV score calculated directly from the original data is 0.19. After training a naïve Bayes classifier from data involving a sensitive feature, the CV score on the predicted classes increases to about 0.34. This shows that Female records are more frequently misclassified to the Low class than Male records; and thus, Female–High individuals are considered to be unfairly treated. This phenomenon is mainly caused by an Occam's razor principle, which is commonly adopted in classifiers. Because infrequent and specific patterns tend to be discarded to generalize observations in data, minority records can be unfairly neglected. Even if the sensitive feature is removed from the training data for a naïve Bayes classifier, the resultant CV score is 0.28, which still shows an unfair treatment for minorities. This is caused by the indirect influence of sensitive features. This event is called by a *red-lining effect* [3], a term that originates from the historical practice of drawing red lines on a map around neighborhoods in which large numbers of minorities are known to dwell. Consequently, simply removing sensitive features is insufficient, and another techniques have to be adopted to correct the unfairness in data mining.

2.2 Three Causes of Unfairness

In this section, we discuss the social fairness in data analysis. Previous works [17,3] have focused on unfairness in the resultant determinations. To look more carefully at the problem of fairness in data mining, we shall examine the underlying causes or sources of unfairness. We suppose that there are at least three possible causes: *prejudice*, *underestimation*, and *negative legacy.*

Before presenting these three causes of unfairness, we must introduce several notations. Here, we discuss supervised learning, such as classification and regression, which is aware of unfairness. Y is a target random variable to be predicted based on the instance values of features. The sensitive variable, S, and non-sensitive variable, X, correspond to sensitive and non-sensitive features, respectively. We further introduce a prediction model $\mathcal{M}[Y|X,S]$, which models a conditional distribution of Y given X and S. With this model and a true distribution over X and S, $\Pr^*[X,S]$, we define

$$\Pr[Y,X,S] = \mathcal{M}[Y|X,S]\Pr^*[X,S]. \tag{1}$$

Applying marginalization and/or Bayes' rule to this equation, we can calculate other distributions, such as $\Pr[Y,S]$ or $\Pr[Y|X]$. We use $\tilde{\Pr}[\cdot]$ to denote sample distributions. $\hat{\Pr}[Y,X,S]$ is defined by replacing a true distribution in (1) with its corresponding sample distribution:

$$\hat{\Pr}[Y,X,S] = \mathcal{M}[Y|X,S]\tilde{\Pr}[X,S], \tag{2}$$

and induced distributions from $\hat{\Pr}[Y,X,S]$ are denoted by using $\hat{\Pr}[\cdot]$.

Prejudice. Prejudice means a statistical dependence between a sensitive variable, S, and the target variable, Y, or a non-sensitive variable, X. There are three types of prejudices: direct prejudice, indirect prejudice, and latent prejudice.

The first type is *direct prejudice*, which is the use of a sensitive variable in a prediction model. If a model with a direct prejudice is used in classification, the classification results clearly depend on sensitive features, thereby generating a database containing *direct discrimination* [17]. To remove this type of prejudice, all that we have to do is simply eliminate the sensitive variable from the prediction model. We then show a relation between this direct prejudice and statistical dependence. After eliminating the sensitive variable, equation (1) can be rewritten as

$$\Pr[Y,X,S] = \mathcal{M}[Y|X]\Pr^*[S|X]\Pr^*[X].$$

This equation states that S and Y are conditionally independent given X, i.e., $\mathrm{Y} \perp\!\!\!\perp \mathrm{S} — \mathrm{X}$. Hence, we can say that when the condition $\mathrm{Y} \not\perp\!\!\!\perp \mathrm{S} — \mathrm{X}$ is not satisfied, the prediction model has a direct prejudice.

The second type is an *indirect prejudice*, which is statistical dependence between a sensitive variable and a target variable. Even if a prediction model lacks a direct prejudice, the model can have an indirect prejudice and can make an unfair determination. We give a simple example. Consider the case that all Y, X, and S are real scalar variables, and these variables satisfy the equations:

$$Y = X + \varepsilon_Y \quad \text{and} \quad S = X + \varepsilon_S,$$

where ε_Y and ε_S are mutually independent random variables. Because $\Pr[Y, X, S]$ is equal to $\Pr[Y|X]\Pr[S|X]\Pr[X]$, these variables satisfy the condition $\mathrm{Y} \perp\!\!\!\perp \mathrm{S}$—X, but do not satisfy the condition $Y \perp\!\!\!\perp S$. Hence, the adopted prediction model does not have a direct prejudice, but may have an indirect prejudice. If the variances of ε_Y and ε_S are small, Y and S become highly correlated. In this case, even if a model does not have a direct prejudice, the determination clearly depends on sensitive information. Such resultant determinations are called indirect discrimination [17] or a red-lining effect [3] as described in section 2.1. To remove this indirect prejudice, we must use a prediction model that satisfies the condition $Y \perp\!\!\!\perp S$.

We next show an index to quantify the degree of indirect prejudice, which is straightforwardly defined as the mutual information between Y and S. However, because a true distribution in equation (1) is unknown, we adopt sample distributions in equation (2) over a given sample set, $\mathcal{D}$:

$$\mathrm{PI} = \sum_{(y,s)\in\mathcal{D}} \hat{\Pr}[y, s] \ln \frac{\hat{\Pr}[y, s]}{\hat{\Pr}[y]\hat{\Pr}[s]}. \tag{3}$$

We refer to this index as a (indirect) *prejudice index* (PI for short). For convenience, the application of the normalization technique for mutual information [21] leads to a *normalized prejudice index* (NPI for short):

$$\mathrm{NPI} = \mathrm{PI}/(\sqrt{\mathrm{H}(Y)\mathrm{H}(S)}), \tag{4}$$

where $\mathrm{H}(\cdot)$ is an entropy function. $\mathrm{PI}/\mathrm{H}(Y)$ is the ratio of information of S used for predicting Y, and $\mathrm{PI}/\mathrm{H}(S)$ is the ratio of information that is exposed if a value of Y is known. This NPI can be interpreted as the geometrical mean of these two ratios. The range of this NPI is $[0, 1]$.

The third type of prejudice is latent prejudice, which is a statistical dependence between a sensitive variable, S, and a non-sensitive variable, X. Consider an example that satisfies the equations:

$$Y = X_1 + \varepsilon_Y, \quad X = X_1 + X_2, \quad \text{and} \quad S = X_2 + \varepsilon_S,$$

where $\varepsilon_Y \perp\!\!\!\perp \varepsilon_S$ and $X_1 \perp\!\!\!\perp X_2$. Clearly, the conditions $\mathrm{Y} \perp\!\!\!\perp \mathrm{S}$—X and $Y \perp\!\!\!\perp S$ are satisfied, but X and S are not mutually independent. This dependence doesn't cause a sensitive information to influence the final determination, but it would be exploited for training learners; thus, this might violate some regulations or laws. Removal of latent prejudice is achieved by making X and Y independent from S simultaneously. Similar to a PI, the degree of a latent prejudice can be quantified by the mutual information between X and S.

Underestimation. Underestimation is the state in which a learned model is not fully converged due to the finiteness of the size of a training data set. Given a learning algorithm that can acquire a prediction model without indirect prejudice, it will make a fair determination if infinite training examples are available. However, if the size of the training data set is finite, the learned classifier may lead to more unfair determinations than that observed in the training sample distribution. Though such determinations are not intentional, they might awake suspicions of unfair treatment. In other words, though the notion of convergence at infinity is appropriate in a mathematical sense, it might not be in a social sense. We can quantify the degree of underestimation by assessing the resultant difference between the training sample distribution over $\mathcal{D}$, $\tilde{\Pr}[\cdot]$, and the distribution induced by a model, $\hat{\Pr}[\cdot]$. Along this line, we define the *underestimation index* (UEI) using the Hellinger distance:

$$\mathrm{UEI} = \sqrt{\frac{1}{2} \sum_{y,s \in \mathcal{D}} \left(\sqrt{\hat{\Pr}[y,s]} - \sqrt{\tilde{\Pr}[y,s]} \right)^2} = \sqrt{1 - \sum_{y,s \in \mathcal{D}} \sqrt{\hat{\Pr}[Y,S] \tilde{\Pr}[Y,S]}}. \quad (5)$$

Note that we did not adopt the KL-divergence because it can be infinite and this property is inconvenient for an index.

Negative Legacy. Negative legacy is unfair sampling or labeling in the training data. For example, if a bank has been refusing credit to minority people without assessing them, the records of minority people are less sampled in a training data set. A sample selection bias is caused by such biased sampling depending on the features of samples. It is known that the problem of a sample selection bias can be avoided by adopting specific types of classification algorithms [24]. However, it is not easy to detect the existence of a sample selection bias only by observing training data. On the other hand, if a bank has been unfairly rejecting the loans of the people who should have been approved, the labels in the training data would become unfair. This problem is serious because it is hard to detect and correct. However, if other information, e.g., a small-sized fairly labeled data set, can be exploited, this problem can be corrected by techniques such as transfer learning [10].

Regulations or laws that demand the removal of latent prejudices are rare. We investigate UEIs in the experimental sections of this paper, but we don't especially focus on underestimation. As described above, avoiding a negative legacy can be difficult if no additional information is available. We therefore focus on the development of a method to remove indirect prejudice.

3 Prejudice Removal Techniques

We here propose a technique to reduce indirect prejudice. Because this technique is implemented as a regularizer, which we call a prejudice remover, it can be applied to wide variety of prediction algorithms with probabilistic discriminative models.

3.1 General Framework

We focused on classification and built our regularizers into logistic regression models. Y, X, and S are random variables corresponding to a class, non-sensitive features, and a sensitive feature, respectively. A training data set consists of the instances of these random variables, i.e., $\mathcal{D} = \{(y, \mathbf{x}, s)\}$. The conditional probability of a class given non-sensitive and sensitive features is modeled by $\mathcal{M}[Y|X, S; \boldsymbol{\Theta}]$, where $\boldsymbol{\Theta}$ is the set of model parameters. These parameters are estimated based on the maximum likelihood principle; that is, the parameters are tuned so as to maximize the log-likelihood:

$$\mathcal{L}(\mathcal{D}; \boldsymbol{\Theta}) = \sum_{(y_i, \mathbf{x}_i, s_i) \in \mathcal{D}} \ln \mathcal{M}[y_i | \mathbf{x}_i, s_i; \boldsymbol{\Theta}]. \tag{6}$$

We adopted two types of regularizers. The first regularizer is a standard one to avoid over-fitting. We used an L_2 regularizer $\|\boldsymbol{\Theta}\|_2^2$. The second regularizer, $R(\mathcal{D}, \boldsymbol{\Theta})$, is introduced to enforce fair classification. We designed this regularizer to be easy to implement and to require only modest computational resources. By adding these two regularizers to equation (6), the objective function to minimize is obtained:

$$-\mathcal{L}(\mathcal{D}; \boldsymbol{\Theta}) + \eta \mathrm{R}(\mathcal{D}, \boldsymbol{\Theta}) + \frac{\lambda}{2} \|\boldsymbol{\Theta}\|_2^2, \tag{7}$$

where λ and η are positive regularization parameters.

We dealt with a classification problem in which the target value Y is binary $\{0, 1\}$, X takes a real vectors, $\mathbf{x}$, and S takes a discrete value, s, in a domain $\mathcal{S}$. We used a logistic regression model as a prediction model:

$$\mathcal{M}[y|\mathbf{x}, s; \boldsymbol{\Theta}] = y \sigma(\mathbf{x}^\top \mathbf{w}_s) + (1 - y)(1 - \sigma(\mathbf{x}^\top \mathbf{w}_s)), \tag{8}$$

where $\sigma(\cdot)$ is a sigmoid function, and the parameters are weight vectors for $\mathbf{x}$, $\boldsymbol{\Theta} = \{\mathbf{w}_s\}_{s \in \mathcal{S}}$. Note that a constant term is included in $\mathbf{x}$ without loss of generality. We next introduce a regularizer to reduce the indirect prejudice.

3.2 Prejudice Remover

A *prejudice remover* regularizer directly tries to reduce the prejudice index and is denoted by R_{PR}. Recall that the prejudice index is defined as

$$\mathrm{PI} = \sum_{Y,S} \hat{\Pr}[Y, S] \ln \frac{\hat{\Pr}[Y, S]}{\hat{\Pr}[S] \hat{\Pr}[Y]} = \sum_{X,S} \tilde{\Pr}[X, S] \sum_{Y} \mathcal{M}[Y|X, S; \boldsymbol{\Theta}] \ln \frac{\hat{\Pr}[Y, S]}{\hat{\Pr}[S] \hat{\Pr}[Y]}.$$

$\sum_{X,S} \tilde{\Pr}[X, S]$ can be replaced with $(1/|\mathcal{D}|) \sum_{(\mathbf{x}_i, s_i) \in \mathcal{D}}$, and then the scaling factor, $1/|\mathcal{D}|$, can be omitted. The argument of the logarithm can be rewritten as $\hat{\Pr}[Y|s_i] / \hat{\Pr}[Y]$, by reducing $\hat{\Pr}[S]$. We obtain

$$\sum_{(\mathbf{x}_i, s_i) \in \mathcal{D}} \sum_{y \in \{0,1\}} \mathcal{M}[y|\mathbf{x}_i, s_i; \boldsymbol{\Theta}] \ln \frac{\hat{\Pr}[y|s_i]}{\hat{\Pr}[y]}.$$

The straight way to compute $\hat{\Pr}[y|s]$ is to marginalize $\mathcal{M}[y|X,s;\boldsymbol{\Theta}]\Pr^*[X|s]$ over X. However, if the domain of X is large, this marginalization is computationally heavy. We hence take an approach by which this marginalization is replaced with a sample mean. More specifically, this marginalization is formulated by

$$\hat{\Pr}[y|s] = \int_{\mathrm{dom}(X)} \Pr^*[X|s]\mathcal{M}[y|X,s;\boldsymbol{\Theta}]dX,$$

where $\mathrm{dom}(X)$ is the domain of X. We approximated this formula by the following sample mean:

$$\hat{\Pr}[y|s] \approx \frac{\sum_{(\mathbf{x}_i,s_i)\in\mathcal{D}\text{ s.t. }s_i=s} \mathcal{M}[y|\mathbf{x}_i,s;\boldsymbol{\Theta}]}{|\{(\mathbf{x}_i,s_i)\in\mathcal{D}\text{ s.t. }s_i=s\}|}. \tag{9}$$

Similarly, we approximated $\hat{\Pr}[y]$ by

$$\hat{\Pr}[y] \approx \frac{\sum_{(\mathbf{x}_i,s_i)\in\mathcal{D}} \mathcal{M}[y|\mathbf{x}_i,s_i;\boldsymbol{\Theta}]}{|\mathcal{D}|}. \tag{10}$$

Note that in our preliminary work [12], we took the approach of replacing X with $\bar{x}_s$, which is a sample mean vector of $\mathbf{x}$ over a set of training samples whose corresponding sensitive feature is equal to s. However, we unfortunately failed to obtain good approximations by this approach.

Finally, the prejudice remover regularizer $\mathrm{R}_{\mathsf{PR}}(\mathcal{D},\boldsymbol{\Theta})$ is

$$\sum_{(\mathbf{x}_i,s_i)\in\mathcal{D}} \sum_{y\in\{0,1\}} \mathcal{M}[y|\mathbf{x}_i,s_i;\boldsymbol{\Theta}] \ln \frac{\hat{\Pr}[y|s_i]}{\hat{\Pr}[y]}, \tag{11}$$

where $\hat{\Pr}[y|s]$ and $\hat{\Pr}[y]$ are equations (9) and (10), respectively. This regularizer becomes large when a class is determined mainly based on sensitive features; thus, sensitive features become less influential in the final determination. In the case of logistic regression, the objective function (7) to minimize is rewritten as

$$\sum_{(y_i,\mathbf{x}_i,s_i)} \ln \mathcal{M}[y_i|\mathbf{x}_i,s_i;\boldsymbol{\Theta}] + \eta\,\mathrm{R}_{\mathsf{PR}}(\mathcal{D},\boldsymbol{\Theta}) + \frac{\lambda}{2}\sum_{s\in\mathcal{S}} \|\mathbf{w}_s\|_2^2, \tag{12}$$

where $\mathcal{M}[y|\mathbf{x},s;\boldsymbol{\Theta}]$ is equation (8) and $\mathrm{R}_{\mathsf{PR}}(\mathcal{D},\boldsymbol{\Theta})$ is equation (11). In our experiment, parameter sets are initialized by applying standard logistic regression to training sets according to the values of a sensitive feature, and this objective function is minimized by a conjugate gradient method. After this optimization, we obtain an optimal parameter set, $\{\mathbf{w}_s^*\}$.

The probability of $Y=1$ given a sample without a class label, $(\mathbf{x}_{\mathrm{new}}, s_{\mathrm{new}})$ can be predicted by

$$\Pr[Y{=}1|\mathbf{x}_{\mathrm{new}},s_{\mathrm{new}};\{\mathbf{w}_s^*\}] = \sigma(\mathbf{x}_{\mathrm{new}}^\top \mathbf{w}_{s_{\mathrm{new}}}^*).$$

4 Experiments

We compared our method with Calders and Verwer's method on the real data set used in their previous study [3].

4.1 Calders-Verwer's 2-naïve-Bayes

We briefly introduce Calders and Verwer's 2-naïve-Bayes method (CV2NB for short), which was found to be the best of three methods in the previous study using the same dataset [3]. The generative model of this method is

$$\Pr[Y, \mathbf{X}, S] = \mathcal{M}[Y, S] \prod_i \mathcal{M}[X_i | Y, S]. \tag{13}$$

$\mathcal{M}[X_i|Y,S]$ models a conditional distribution of X_i given Y and S, and the parameters of these models are estimated in a similar way as in the estimation of parameters of a naïve Bayes model. $\mathcal{M}[Y,S]$ models a joint distribution Y and S. Because Y and S are not mutually independent, the final determination might be unfair. While each feature depends only on a class in the case of the original naïve Bayes, every non-sensitive feature, X_i, depends on both Y and S in the case of CV2NB. $\mathcal{M}[Y,S]$ is then modified so that the resultant CV score approaches zero. Note that we slightly changed this algorithm as described in [12], because the original algorithm may fail to stop.

4.2 Experimental Conditions

We summarize our experimental conditions. We tested a previously used real data set [3], as shown in section 2.1. This set includes 16281 data in an adult.test file of the Adult / Census Income distributed at the UCI Repository [7]. The target variable indicates whether or not income is larger than 50M dollars, and the sensitive feature is gender. Thirteen non-sensitive features were discretized by the procedure in the original paper. In the case of the naïve Bayes, parameters of models, $\mathcal{M}[X_i|Y,S]$, are estimated by a MAP estimator with multinomial distribution and Dirichlet priors. In our case of logistic regression, discrete variables are represented by 0/1 dummy variables coded by a so-called 1-of-K scheme. The regularization parameter for the L_2 regularizer, λ, is fixed to 1, because the performance of pure logistic regression was less affected by this parameter in our preliminary experiments. We tested six methods: logistic regression with a sensitive feature (LR), logistic regression without a sensitive feature (LRns), logistic regression with a prejudice remover regularizer (PR), naïve Bayes with a sensitive feature (NB), naïve Bayes without a sensitive feature (NBns), and Calders and Verwer's 2-naïve-Bayes (CV2NB). We show the means of the statistics obtained by the five-fold cross-validation.

4.3 Experimental Results

Table 1 shows accuracies (Acc), NPI and UEI in section 2, and CV scores (CVS). MI denotes mutual information between sample labels and predicted labels; NMI

Table 1. A summary of experimental results

method	Acc	NMI	NPI	UEI	CVS	PI/MI
LR	0.851	0.267	5.21E-02	0.040	0.189	2.10E-01
LRns	0.850	0.266	4.91E-02	0.039	0.184	1.99E-01
PR η=5	0.842	0.240	4.24E-02	0.088	0.143	1.91E-01
PR η=15	0.801	0.158	2.38E-02	0.212	0.050	1.62E-01
PR η=30	0.769	0.046	1.68E-02	0.191	0.010	3.94E-01
NB	0.822	0.246	1.12E-01	0.068	0.332	4.90E-01
NBns	0.826	0.249	7.17E-02	0.043	0.267	3.11E-01
CV2NB	0.813	0.191	3.64E-06	0.082	-0.002	2.05E-05

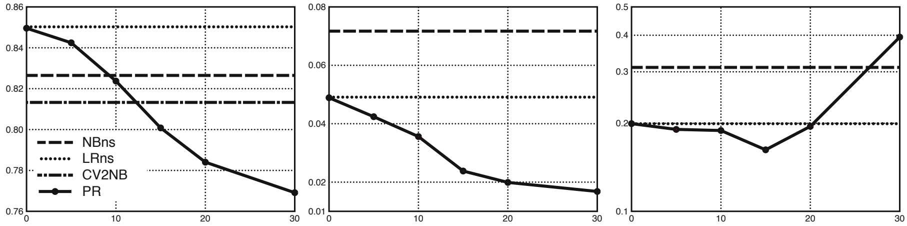
NOTE: $\langle n_1 \rangle \mathrm{E} \langle n_2 \rangle$ denotes $n_1 \times 10^{n_2}$. L_2 regularizer: $\lambda = 1$.

was obtained by normalizing this MI in a process similar to NPI. PI/MI quantifies a prejudice index that was sacrificed by obtaining a unit of information about the correct label. This can be used to measure the efficiency in the trade-off between prediction accuracy and prejudice removal. The smaller PI/MI value indicates higher efficiency in this trade-off.

We first compare the performance of our method with that of baselines in Table 1. Compared with NBns, our method was superior both in accuracy and NPI at $\eta = 5$; and hence, ours was superior in the efficiency index, PI/MI. When comparing LRns, the prejudice in decisions was successfully removed by our prejudice remover in exchange for the prediction accuracy. We next moved on to the influence of the parameter, η, which controls the degree of prejudice removal. We expected that the larger the η, the more prejudice would be removed, whereas accuracy might be sacrificed. According to Table 1, as η increased, our PR generally become degraded in accuracy.

To further investigate the change of performance depending on this parameter η, we demonstrated the variations in accuracy (Acc), normalized prejudice index (NPI), and the trade-off efficiency between accuracy and prejudice removal (PI/MI) in Figure 1. We focus on our PR method. The increase of η generally damaged accuracy because the prejudice remover regularizer is designed to remove prejudice by sacrificing accuracy in prediction. This effect was observed by the increase in NPI. The peak in trade-off efficiency was observed at $\eta = 15$. More prejudice could be removed by increasing η, but the accuracy in prediction was fairly damaged.

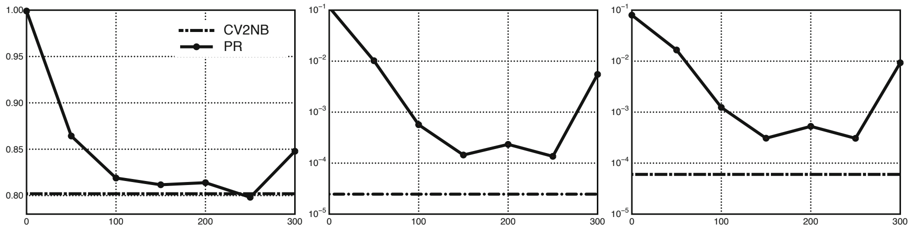
We next compared our PR with other methods. By observing Figure 1(c), our PR demonstrated better performance in trade-offs between accuracy and prejudice removal than the NBns. When compared to the baseline LRns, more prejudice was successfully removed by increasing η. The Figure 1(a) showed that this was achieved by sacrificing the prediction accuracy. The efficiencies in the trade-offs of our PR was better than those of LRns if η ranged between 0 and 20. The performance of CV2NB was fairly good, and our PR was inferior to it except for accuracy at the lower range of η.



(a) Acc (b) NPI (c) PI/MI

Fig. 1. The change in performance for the Adult / Census Income data according to η

NOTE: Horizontal axes represent the parameter η, and vertical axes represent statistics in each subtitle. Solid, chain, dotted, and broken lines indicate the statistics of PR, CV2NB, LRns, and NBns, respectively. Larger Acc values indicate better performance, and smaller NPI and PI/MI values indicate better performance. NPI and PI/MI of CV2NB were out of the bounds of these charts and are properly noted in Table 1.



(a) Acc (b) NPI (c) PI/MI

Fig. 2. The change in performance for our synthetic data according to η

NOTE: The meanings of axes and line styles are the same as in Figure 1.

To show how the difference in the prejudice removal between CV2NB and PR is brought about by the ability of our method to take into account the difference in influence of different features on sensitive information, we applied our PR to a synthetic data set. To synthesize data, ϵ_i was sampled from the normal distribution $\mathcal{N}(0, 1)$, and $s_i \in \{0, 1\}$ was sampled uniformly at random. The first feature $x_{ai} = \epsilon_i$, and the second feature $x_{bi} = 1 + \epsilon_i$ if $s_i = 1$; otherwise $x_{bi} = -1+\epsilon_i$. The class y_i was set to 0 if $x_{ai}+x_{bi} < 0$; otherwise 1. We generated 20 000 samples and applied CV2NB and our PR by changing η from 0 to 300. Because x_{ai} and x_{bi} are equivalent up to bias, these two features are comparable in usefulness for class prediction. The first feature, x_{ai}, is independent from s_i, while the second feature, x_{bi}, depends on s_i.

We showed the change in three indexes, accuracy, NPI, and PI/MI, on independently generated test data according to the parameter η in Figure 1. Unfortunately, results became unstable if η is larger than 200 because the objective function (12) has many local minima for large η. However, when comparing the

Table 2. The learned weight vectors $\mathbf{w}_0$ and $\mathbf{w}_1$ in equation (8)

	$\mathbf{w}_0$	$\mathbf{w}_1$
$\eta = 0$	[11.3, 11.3 ,−0.0257]	[11.3, 11.4 ,0.0595]
$\eta = 150$	[55.3,−53.0, −53.6]	[56.1,−54.1, 53.6]

NOTE: The first, second, and third elements of $\mathbf{w}_s$ were weights for the first feature, x_{ai}, the second feature, x_{bi}, and a bias constant, respectively.

results in Table 1 with those in this figure, the differences in NPI derived by CV2NB and PR became much smaller.

To exemplify the reason for these differences, we then showed the learned weight vectors $\mathbf{w}_0$ and $\mathbf{w}_1$ in equation (8) in Table 2. By observing the weights more carefully, the weights for x_{ai} and x_{bi} were roughly equal when $\eta = 0$. However, when $\eta = 150$, the absolute values of weights for x_{bi} were smaller than those for x_{ai}. This indicates that to remove prejudice, our PR tries to ignore features that depend on a sensitive feature. Therefore, if there are features that are useful for classification and additionally independent from a sensitive feature, our PR can remove prejudice effectively. In other words, our method is designed to learn a classification model by taking into account difference in influence of different features on sensitive information. On the other hand, according to the generative model (13), CV2NB treats all features equally and simply modifies the $\mathcal{M}[Y, S]$ for removing prejudice. Therefore, CV2NB cannot learn a model that reflects such differences.

This difference would cause the following effect in practical use. When considering a case of credit scoring, because CV2NB treats all features equally, scores of all individuals who are in a sensitive state would be raised equally. However, the repayment capacities of these individuals are certainly unequal, and our method can change credit scoring by taking into account individuals' repayment capacity. On the other hand, if the repayment capacities of all individuals in a sensitive state are nearly equal, our method cannot reduce prejudice without degrading prediction accuracy. However, CV2NB can remove prejudice independently of the states of individuals' repayment capacity. Note that fair decision-making that takes into account the differences in effects of features has also been discussed in [13,23].

In summary, our PR could successfully reduce indirect prejudice when compared with baseline methods. Our method is inferior to CV2NB in its efficiency of prejudice removal, but it can learn a classification rule by taking into account the difference in influence of different features on sensitive information. Additionally, our framework has the advantage that it can be applied to any probabilistic discriminative classifier.

5 Related Work

Several analytic techniques that are aware of fairness or discrimination have recently received attention. Pedreschi et al. emphasized the unfairness in association rules whose consequents include serious determinations [17].They advocated the notion of *α-protection*, which is the condition that association rules were fair. Given a rule whose consequent exhibited determination is disadvantageous to individuals, it would be unfair if the confidence of the rule substantially increased by adding a condition associated with a sensitive feature to the antecedent part of the rule. The α-protection constrains the rule so that the ratio of this increase is at most α. They also suggested the notions of *direct discrimination* and *indirect discrimination*. A direct discriminatory rule directly contains a sensitive condition in its antecedent, and while an indirect discriminatory rule doesn't directly contain a sensitive condition, the rule is considered to be unfair in the context of background knowledge that includes sensitive information. Their work has since been extended [18]. Various kinds of indexes for evaluating discriminatory determinations were proposed and their statistical significance has been discussed. A system for finding such unfair rules has been proposed [20].

Calders and Verwer proposed several methods to modify naïve Bayes for enhancing fairness as described in section 4.1 [3]. Kamiran et al. developed algorithms for learning decision trees while taking fairness consideration [11]. When choosing features to divide training examples at non-leaf nodes of decision trees, their algorithms take care of the information gain regarding sensitive information as well as about target decisions. Additionally, the labels at leaf nodes are changed so as to avoid unfair decisions.

Luong et al. proposed a notion of situation testing, wherein a determination is considered unfair if different determinations are made for two individuals all of whose features are equal except for sensitive ones [13]. Such unfairness was detected by comparing the determinations for records whose sensitive features are different, but are neighbors in non-sensitive feature space. If a target determination differs, but non-sensitive features are completely equal, then a target variable depends on a sensitive variable. Therefore, this situation testing has connection to our indirect prejudice.

Dwork et al. argued a data transformation for the purpose of exporting data while keeping aware of fairness [5]. A data set held by a data owner is transformed and passed to a vendor who classifies the transformed data. The transformation preserves the neighborhood relations of data and the equivalence between the expectations of data mapped from sensitive individuals and from non-sensitive ones. In a sense that considering the neighborhood relations, this approach is related to the above notion of situation testing. Because their proposition 2.2 implies that the classification results are roughly independent from the membership in a sensitive group, their approach has relation to our idea of prejudice.

Žliobaitė et al. discussed handling conditional discrimination [23]. They considered the case where even if the difference between probabilities of receiving advantageous judgment given different values of sensitive features, some extent of the difference can be explained based on the values of non-sensitive features. For example, even though females are less frequently admitted to a university than males, this decision is considered as fair if this is due to the fact that females tend to try more competitive programs. They proposed a sampling technique to remove the unfair information from training samples while excluding such explainable factors.

In a broad sense, fairness-aware learning is related to causal inference [16], because the final decision becomes unfair if the decision depends on a sensitive status. Fairness in data mining can be interpreted as a sub-notion of legitimacy, which means that models can be deployed in the real world [19]. Gondek and Hofmann devised a method for finding clusters that were not relevant to a given grouping [8]. If a given grouping contains sensitive information, this method can be used for clustering data into fair clusters. Independent component analysis might be used to maintain the independence between features [9].

The removal of prejudice is closely related to privacy-preserving data mining [1], which is a technology for mining useful information without exposing individual private records. The privacy protection level is quantified by mutual information between the public and private realms [22]. In our case, the degree of indirect prejudice is quantified by mutual information between classification results and sensitive features. Due to the similarity of these two uses of mutual information, the design goal of fairness-aware learning can be considered the protection of sensitive information when exposing classification results. In our case, the leaked information is quantified by mutual information, but other criteria for privacy, such as differential privacy [14], might be used for the purpose of maintaining fairness.

Techniques of cost-sensitive learning [6] might be helpful for addressing underestimation problems.

As described in section 2.2, the problem of negative legacy is closely related to transfer learning. Transfer learning is “the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task” [10]. Among many types of transfer learning, the problem of a sample selection bias [24] would be related to the negative legacy problem. Sample selection bias means that the sampling is not at random, but biased depending on some feature values of data. Another related approach to transfer learning is weighting samples according the degree of usefulness for the target task [4]. Using these approaches, if given a small amount of fairly labeled data, other data sets that might be unfairly labeled would be correctly processed.

6 Conclusions and Future Work

The contributions of this paper are as follows. First, we proposed three causes of unfairness: prejudice, underestimation, and negative legacy. Prejudice refers to the dependence between sensitive information and the other information, either directly or indirectly. We further classified prejudice into three types and developed a way to quantify them by mutual information. Underestimation is the state in which a classifier has not yet converged, thereby producing more unfair determinations than those observed in a sample distribution. Negative legacy is the problem of unfair sampling or labeling in the training data. Second, we developed techniques to reduce indirect prejudice. We proposed a prejudice remover regularizer, which enforces a classifier's independence from sensitive information. Our methods can be applied to any algorithms with probabilistic discriminative models and are simple to implement. Third, we showed experimental results of logistic regressions with our prejudice remover regularizer. The experimental results showed the effectiveness and characteristics of our methods.

Research on fairness-aware learning is just beginning, and there are many problems yet to be solved: for example, the definition of fairness in data analysis, measures for fairness, and maintaining other types of laws or regulations. The types of analytic methods are severely limited at present. Our method can be easily applied to regression, but fairness-aware clustering and ranking methods are also needed. Because of the lack of convexity of the objective function, our method is occasionally trapped by local minima. To avoid this, we plan to try other types of independence indexes, such as kurtosis, which has been used for independent component analysis. If a sensitive feature is a multivariate variable whose domain is large or is a real variable, our current prejudice remover cannot be applied directly; these limitations must be overcome.

The use of data mining technologies in our society will only become greater and greater. Unfortunately, their results can occasionally damage people's lives [2]. On the other hand, data analysis is crucial for enhancing public welfare. For example, exploiting personal information has proved to be effective for reducing energy consumption, improving the efficiency of traffic control, preventing infectious diseases, and so on. Consequently, methods of data exploitation that do not damage people's lives, such as fairness/discrimination-aware learning, privacy-preserving data mining, or adversarial learning, together comprise the notion of *socially responsible mining*, which it should become an important concept in the near future.

Acknowledgments. We wish to thank Dr. Sicco Verwer for providing detail information about his work and to thank the PADM2011 workshop organizers and anonymous reviewers for their valuable comments. This work is supported by MEXT/JSPS KAKENHI Grant Number 16700157, 21500154, 22500142, 23240043, and 24500194, and JST PRESTO 09152492.

References

1. Aggarwal, C.C., Yu, P.S. (eds.): Privacy-Preserving Data Mining: Models and Algorithms. Springer (2008)
2. Boyd, D.: Privacy and publicity in the context of big data. In: Keynote Talk of The 19th Int'l Conf. on World Wide Web (2010)
3. Calders, T., Verwer, S.: Three naive bayes approaches for discrimination-free classification. Data Mining and Knowledge Discovery 21, 277–292 (2010)
4. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proc. of the 24th Int'l Conf. on Machine Learning, pp. 193–200 (2007)
5. Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.: Fairness through awareness. arxiv.org:1104.3913 (2011)
6. Elkan, C.: The foundations of cost-sensitive learning. In: Proc. of the 17th Int'l Joint Conf. on Artificial Intelligence, pp. 973–978 (2001)

7. Frank, A., Asuncion, A.: UCI machine learning repository. School of Information and Computer Sciences, University of California, Irvine (2010), http://archive.ics.uci.edu/ml
8. Gondek, D., Hofmann, T.: Non-redundant data clustering. In: Proc. of the 4th IEEE Int'l Conf. on Data Mining, pp. 75–82 (2004)
9. Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. Wiley-Interscience (2001)
10. NIPS workshop — inductive transfer: 10 years later (2005), http://iitrl.acadiau.ca/itws05/
11. Kamiran, F., Calders, T., Pechenizkiy, M.: Discrimination aware decision tree learning. In: Proc. of the 10th IEEE Int'l Conf. on Data Mining, pp. 869–874 (2010)
12. Kamishima, T., Akaho, S., Sakuma, J.: Fairness-aware learning through regularization approach. In: Proc. of The 3rd IEEE Int'l Workshop on Privacy Aspects of Data Mining, pp. 643–650 (2011)
13. Luong, B.T., Ruggieri, S., Turini, F.: k-NN as an implementation of situation testing for discrimination discovery and prevention. In: Proc. of the 17th Int'l Conf. on Knowledge Discovery and Data Mining, pp. 502–510 (2011)
14. Nissim, K.: Private data analysis via output perturbation. In: Aggarwal, C.C., Yu, P.S. (eds.) Privacy-Preserving Data Mining: Models and Algorithms, ch. 4. Springer (2008)
15. Pariser, E.: The Filter Bubble: What The Internet Is Hiding From You. Viking (2011)
16. Pearl, J.: Causality: Models, Reasoning and Inference, 2nd edn. Cambridge University Press (2009)
17. Pedreschi, D., Ruggieri, S., Turini, F.: Discrimination-aware data mining. In: Proc. of the 14th Int'l Conf. on Knowledge Discovery and Data Mining (2008)
18. Pedreschi, D., Ruggieri, S., Turini, F.: Measuring discrimination in socially-sensitive decision records. In: Proc. of the SIAM Int'l Conf. on Data Mining, pp. 581–592 (2009)
19. Perlich, C., Kaufman, S., Rosset, S.: Leakage in data mining: Formulation, detection, and avoidance. In: Proc. of the 17th Int'l Conf. on Knowledge Discovery and Data Mining, pp. 556–563 (2011)
20. Ruggieri, S., Pedreschi, D., Turini, F.: DCUBE: Discrimination discovery in databases. In: Proc of The ACM SIGMOD Int'l Conf. on Management of Data, pp. 1127–1130 (2010)
21. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2002)
22. Venkatasubramanian, S.: Measures of anonimity. In: Aggarwal, C.C., Yu, P.S. (eds.) Privacy-Preserving Data Mining: Models and Algorithms, ch. 4. Springer (2008)
23. Žliobaitė, I., Kamiran, F., Calders, T.: Handling conditional discrimination. In: Proc. of the 11th IEEE Int'l Conf. on Data Mining (2011)
24. Zadrozny, B.: Learning and evaluating classifiers under sample selection bias. In: Proc. of the 21st Int'l Conf. on Machine Learning, pp. 903–910 (2004)

A Live Comparison of Methods for Personalized Article Recommendation at Forbes.com

Evan Kirshenbaum[1], George Forman[1], and Michael Dugan[2]

[1] HP Labs, Palo Alto, CA, USA
[2] Forbes Media, New York, NY, USA

Abstract. We present the results of a multi-phase study to optimize strategies for generating personalized article recommendations at the Forbes.com web site. In the first phase we compared the performance of a variety of recommendation methods on historical data. In the second phase we deployed a live system at Forbes.com for five months on a sample of 82,000 users, each randomly assigned to one of 20 methods. We analyze the live results both in terms of click-through rate (CTR) and user session lengths. The method with the best CTR was a hybrid of collaborative-filtering and a content-based method that leverages Wikipedia-based concept features, post-processed by a novel Bayesian remapping technique that we introduce. It both statistically significantly beat decayed popularity and increased CTR by 37%.

Keywords: personalization, recommender systems, collaborative filtering, content analysis, live user trial.

1 Introduction

We performed an extensive study on generating personalized recommendations of articles on the Forbes.com web site. Of the many algorithms available, which is the best to deploy in practice? While each research paper on the topic forwards its own opinion, the answer is certainly that it depends on the specific situation. A study done on movie recommendation might draw different conclusions than one conducted on news articles, which have a much shorter half-life and suggest a stronger recency factor in scoring. Even a study conducted specifically on news recommendation may draw different conclusions than one specifically targeting a website like Forbes.com, which includes magazine articles and other long-lived content, such as how-to articles and profiles of top colleges and business people. Even a typical questionnaire-style study with a few volunteer Forbes users is unlikely to generalize well to real use by live users. In short, there is no substitute for trying a variety of methods *in situ* for selecting the best method(s) for one's situation. That said, only a small number of methods can be tested in a live trial, so they should be from among the most likely to succeed.

We conducted our study in two phases. In the first phase, we used a historical dataset of 8.5 million article URL clicks by 1 million unique de-identified users in order to quickly test a wide variety of recommendation methods, including

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 51–66, 2012.

variants and parameterizations. From this study we determined a short list to evaluate in live testing in the second phase. Evaluation with historical datasets is reproducible and convenient to test many methods, but it has a variety of shortcomings and may not generalize to real-world performance [24,4,22]. If an algorithm gives top scores to articles not in the user's history, they may be more desirable to the user than the recorded clicks...or less desirable—only live testing can distinguish. In the worst case, the historical data represents less what users actually like than which links were available or promoted on the home page at the time the user visited, which can have a very strong effect on popularity [16].

In the second phase—the primary focus of this paper—we conducted a five-month live trial on the Forbes.com web site involving 2.1 million URL clicks by a sample of 82,000 de-identified users, each assigned randomly to one of twenty competing methods. Our goals and constraints included the following: It must deal with only de-identified user numbers, not requiring user demographics or any personal profile information. It should be scalable to 100,000 users, 100,000 articles, and dozens of competing methods running simultaneously with sub-second latency for months. The study was made viable by minimizing the changes needed by Forbes.

The winning method involves a hybrid of item-item collaborative filtering and a content-based TF·IDF method, including Wikipedia-based features and a novel Bayesian score remapping technique that takes into account the models of other users that read—or were inferred to have chosen not to read—the article. We also evaluated pure versions of each method as well as various lesions, in order to determine the usefulness of the different aspects. Because our goal was to seek the best methods rather than to restrict ourselves to comparing only variants of one technique, we have good confidence that the best method we found is actually quite strong for our situation. Live trials of this kind and scale are relatively scarce in the literature. This work demonstrates a successful pattern for live trial research for researchers who do not themselves work for a major content company with a large user population.

Section 2 describes the recommender algorithms we considered in both Phase I and Phase II, including novel techniques for score remapping and Wikipedia-based concept features. Section 3 describes the live trial, our experiment protocol, and the results, including lesion variants to determine how much certain features of the best hybrid are contributing. Section 4 discusses the findings and lists some of the lessons learned. Section 5 discusses related work, and Section 6 concludes and offers future directions.

Further details and discussion of topics omitted for reasons of space may be found in a longer technical report version of this article available from HP Labs.

2 Recommendation Methods

In the first phase of the experiment, we analyzed a historical snapshot, provided by Forbes Media, of 8.5 million de-identified user visits to the Forbes.com website in order to determine promising candidate recommendation methods for the live trial in the second phase. The snapshot was split into a training set consisting of 8.2 million visits to articles published from May, 2010, through November, 2010, and a testing set consisting of 285,000 visits to articles published in December, 2010, by 117,000 unique users. As there was no possibility of actually making recommendations and due to other constraints imposed by the data set—most notably that the visits were not timestamped—the score given each candidate recommendation method was based on a determination, for each visit, as to whether the article visited would have been one of the top-scoring articles published within five days of the publication date of the visited article (reasoning that the "relevancy window" for an article is approximately five days and so the visit was likely within five days after publication and the competing articles were published fewer than five days before the visit).

In the first phase we ran approximately 15,000 trials, each testing a particular parameterization of a particular scoring method. Among the content-based methods, we tested variants of Naïve Bayes and TF·IDF, each with a variety of feature set combinations and other parameters, such as variants on how exactly Inverse Document Frequency is computed. Among the feature sets tested were unigram words-only and words+bigrams from the text of the article, from its title, and/or from its URL. We also included four classes of features generated by two Wikipedia-based *concept extractors* developed previously, the best of which we describe briefly in Sect. 2.2. Among collaborative filtering methods [24], we tested Item-Item collaborative filtering and User-User collaborative filtering, parameterized by several similarity metrics: Cosine, Jaccard, Euclidean, Pearson correlation, and conditional probability in each direction. We also tested the one-class collaborative filtering methods weighted Alternating Least Squares (ALS) and sampled ALS ensembles, whose regularized matrix factorizations seek to learn latent factors across the entire population [19]. As our control method, we investigated using article popularity. We also investigated linear- and non-linear combinations of article popularity with various scoring methods, as well as a novel score remapping technique, which we discuss next.

2.1 Bayesian Score Remapping Technique

For this experiment, we developed a new technique, based on Bayes' Rule, that adjusts a score from any underlying scoring method by estimating the likelihood that a user with that particular score would actually be interested enough to visit the article. In this way, we are able to distinguish between articles that are broadly popular (i.e., that empirically appeal to users whose underlying scores are relatively low) and those that are narrowly popular (i.e., that only appeal to users with high underlying scores), allowing us to reorder our estimates of a user's rank interest from what the underlying method would recommend.

To accomplish this, for each article we keep track of two score distributions, modeled as normal distributions. The first distribution contains the scores for this article of all users who have read it, based on the current underlying model for those users. This allows us to compute the conditional probability that a user who was interested to read the article has a given score for it. The second distribution similarly contains scores for the article, but in this case, the scores are for users who did not read the article. Rather than take the scores of all such users, we use the scores of users who were active on the site shortly after the article appeared (the article's "relevancy window") but who did not view the article. These users are taken to be ones who would likely have read the article had they been interested and so are inferred to not have been interested. From this second distribution, we can compute the conditional probability that a user who was not interested in the article has a given score for it. Note that in a live system, users' scores for articles will change frequently and when users view the article they move from the *non-interested-* to the *interested*-distribution, so these distributions need to be dynamically maintained.

By Bayes' Rule,

$$\frac{\Pr[\,A \mid B\,]}{\Pr[\,\overline{A} \mid B\,]} = \frac{\Pr[\,B \mid A\,] \cdot \Pr[\,A\,]}{\Pr[\,B \mid \overline{A}\,] \cdot \Pr[\,\overline{A}\,]} \ . \tag{1}$$

In our context, for a particular underlying score s, the conditional likelihood ratio

$$\begin{aligned} R &= \frac{\Pr[\,\text{interesting} \mid s\,]}{\Pr[\,\text{not interesting} \mid s\,]} && (2)\\ &= \frac{\Pr[\,s \mid \text{interesting}\,] \cdot \Pr[\,\text{interesting}\,]}{\Pr[\,s \mid \text{not interesting}\,] \cdot \Pr[\,\text{not interesting}\,]} && (3)\\ &= \frac{\Pr[\,s \mid \text{interesting}\,]}{\Pr[\,s \mid \text{not interesting}\,]} \cdot \frac{\Pr[\,\text{interesting}\,]}{1 - \Pr[\,\text{interesting}\,]} \ , && (4) \end{aligned}$$

which can be computed directly with our stored distributions. Since

$$\Pr[\,\text{interesting} \mid s\,] = 1 - \Pr[\,\text{not interesting} \mid s\,] \ , \tag{5}$$

$$\Pr[\,\text{interesting} \mid s\,] = \frac{R}{R+1} \ , \tag{6}$$

by use of equation (2) again. Equation (6) is used as an adjusted score indicating the likelihood that a user with a particular underlying score s will be interested in reading the article. This works for any underlying monotonic scoring method.

Some care must be taken when one or both of the distributions for an article contain too few values for the conditional probabilities to be statistically meaningful. In this case, we consider that we do not yet have enough information about the article and do not recommend it.

2.2 Wikiconcept Features

For a prior project we had developed a Wikipedia-based *concept extractor* [12] that takes unstructured text as input and constructs an analysis that includes a set of *concepts*, identified by Wikipedia articles, each associated with a score indicative of the degree to which the text is "about" that concept, with higher scores indicating that the concept is central to the text and lower scores indicating that, although the concept is mentioned in the text, it is relatively incidental.

The details of the concept extractor are beyond the scope of this paper, but roughly, the text is broken up into sentences (or sentence-equivalents) and scanned for the presence of 1–5 word *anchor phrases*, taken from the text displayed on intra-Wikipedia hyperlinks and implicating (often ambiguously) concepts associated with the Wikipedia articles the hyperlinks point to. For example, the anchor phrase “Clinton” is associated with “Bill Clinton” and “Hillary Clinton”, along with “George Clinton”, “Henry Clinton”, “DeWitt Clinton”, “Clinton, Iowa”, and “Clinton County, NY”, each with a prior degree of likelihood based on the number of times the phrase was used within Wikipedia to link to the concept’s article.

The evidence from the detected anchor phrases is passed to an iterative consensus algorithm that determines, based on the evidence and on the conditional likelihood that pairs of Wikipedia articles will both be link targets within a Wikipedia article, the most likely concept referent for each anchor phrase (if any is deemed sufficiently likely). Each concept the extractor knows about is also associated with a set of *categories*, and based on the coocurrence of concepts associated with different categories, one or two (occasionally more) categories are chosen to describe the context of the particular concept. Categories are also inferred directly using a support vector machine trained with non-Wikipedia articles from the Open Directory Project (www.dmoz.org).

The tables used to drive the extractor are generated automatically from a Wikipedia snapshot obtained periodically from Freebase.com. The system used for the live trial includes 6.7 million normalized anchor phrases that impute 3.3 million concepts. The category set includes 913 categories in a hand-crafted hierarchy.

The concept extractor outputs four classes of features for consideration by the content-based methods: (1) detected anchor phrases, annotated by occurrence count, (2) extracted concepts, annotated by centrality score and confidence of detection, (3) recognized categories, associated with categorizer scores, and (4) identified concept/category pairs, annotated by concept centrality and confidence.

2.3 Phase II Methods

Based on the Phase I trials, the details of which must be omitted for space, the clear winner was Bayesian-adjusted TF·IDF, with no IDF component (so just TF), with a logarithmic transformation on the TF counts, and L_2 normalization on the length of the feature vector. Its best-performing feature set included anchor phrases and concepts from the concept extractor (see Sect. 2.2), words from the article title, and words and bigrams from the article URL, but not, interestingly, words or bigrams from the article body. We therefore chose to use TF·IDF, variously parameterized, for our content-based methods. As we also wanted to investigate collaborative filtering in the live trial, we chose the most competitive method: Item-Item collaborative filtering where a user’s score for an unread article U is the conditional probability that other users have read U,

Table 1. Phase II methods

Baseline Methods

1. Decayed popularity
2. Unpersonalized decayed popularity
3. Raw (undecayed) popularity
4. Unadjusted TF·IDF("bag of words" features, no Wikiconcepts features)
5. Unadjusted Item-Item collaborative filtering

Experimental Methods

6. Popularity-adjusted TF
7. Bayesian-adjusted TF
8. 50%-popularity-adjusted Item-Item
9. 10%-popularity-adjusted Item-Item
10. Popularity-adjusted TF/CF hybrid
11. Bayesian-adjusted TF/CF hybrid

Lesion Methods

12. Popularity-adjusted TF (no concepts)
13. Bayesian-adjusted TF (no concepts)
14. Popularity-adjusted TF·IDF (no Wiki features)
15. Bayesian-adjusted TF·IDF (no Wiki features)
16. Unadjusted TF
17. Bayesian-adjusted TF (no recency focus)
18. Popularity-adjusted TF (no negative interest filter)
19. Bayesian-adjusted TF (no negative interest filter)
20. Bayesian-adjusted TF (using CDF)

given that they also read an article R that the user has read, averaged over all articles R the user has read so far.

Based on anticipated traffic volume and experiment duration, we estimated that we could test 20 methods in parallel in Phase II, with each newly-observed user randomly assigned to a recommendation method in a balanced manner, and expect to be able to statistically significantly distinguish between better-performing and worse-performing methods. We considered it important to run all of the methods at the same time so that we could be confident that differences we found were due to the methods themselves and not due to changes in the set of candidate articles (e.g., that one recommendation method had a popular and easily recommendable candidate article not available to another method, were we to use sequential A-B testing).

In Phase II, our recommendation methods consisted of a *scoring function*, which produced a numeric score for each of a set of candidate articles, a set of *filters*, which constrained the set of candidate articles, and a *selection method*, which seclected the articles to recommend based on the computed scores and possibly other information associated with the articles. Unless otherwise mentioned, all Phase II methods included filters that removed from consideration any article that the user had already read or for which the user had at least twice selected a recommended article further down in a recommendation list. Except for Bayesian-adjusted methods, the selection method selected articles with the highest associated scores. For Bayesian-adjusted methods, unless specified, the selection method selected the most recently-published articles from among the 25 highest (adjusted)-scoring articles, with all articles published in the last 48 hours considered to be equally recent. This is an attempt to capture the notion of recency, which is built in for methods that mixed with decayed popularity.

The Phase II methods are listed in Table 1.

Baseline Methods. Five of the twenty methods were chosen as representing state of practice. Method 1 prefers articles that have been visited the most times, with an exponential decay. As with all of the Phase II methods that involved decayed popularity, the smooth decay parameter was chosen such that a visit is worth 10% of one 24 hours later. This decay value was empirically chosen as substantially optimal based on an observational period prior to the start of Phase II, during which we observed user visits but did not make recommendations.

Method 2 is like Method 1, except that popular articles are recommended to a user even if the user has previously read the article or if the user has selected articles below that article in prior recommendation lists—this represents the commonplace, unpersonalized *most popular* lists at many sites. In Method 3, the most popular articles are recommended to a user with no popularity decay.

For the other two baseline methods we chose one content-based method and one collaborative filtering method. The content-based method, Method 4, is unadjusted TF·IDF (including the typical IDF component and logarithmic TF transform), with L_2 normalization over the usual "bag of words" features taken from the article's body, title, and URL. The collaborative filtering method, Method 5, is unadjusted Item-Item collaborative filtering, as described above.

Experimental Methods. The next six methods are the ones that we expected to be serious contenders. Two of them are content-based methods using L_2-normalized, logarithmically-transformed TF over extracted concepts, detected anchor phrases, title words, and URL words and bigrams. The resulting score is, in Method 6, averaged evenly with the score from Method 1 (decayed popularity), while in Method 7, the Bayesian adjustment described in Sect. 2.1 is applied. Even though the Bayesian adjustment outperformed mixing with popularity in the static Phase I trials, we wanted to leave ourselves open to the possibility that in live trials we might get a different result, and so included popularity-adjusted variants to the Bayesian-adjusted methods.

Two experimental methods used a weighted average of Item-Item collaborative filtering with decayed popularity. Method 8 weights them evenly, while Method 9 gives just 10% weight to the popularity component.

In the final two experimental methods, we use both content-based (TF) and collaborative filtering (CF). In Method 10, the scores from TF (as in Method 6), Item-Item collaborative filtering, and decayed popularity are averaged evenly. In Method 11, the scores from TF and Item-Item collaborative filtering are averaged evenly and Bayesian adjustment is applied to the resulting score.

Lesion Methods. Finally, we included nine methods that investigate leaving out or otherwise altering some aspect of one of the experimental methods in order to determine whether that aspect is important. In Methods 12 and 13, we investigate leaving out concepts as features from Methods 6 and 7. While the concept extractor is quite efficient, there is a runtime and software-complexity cost for including the algorithm, and so if it turned out that concepts were needless, omitting them would be an appreciable simplification in feature extraction.

In Methods 14 and 15, we further leave out anchor phrases as features. While the time required to detect these phrases is negligible, if neither concepts nor anchor phrases is required, then there is no need to expend the effort to obtain Wikipedia snapshots and build the required tables. To give these methods the best chance, we chose the best-performing parameterization from Phase I over TF·IDF runs that did not use concept extractor features. The IDF component is included, and the features are title words, URL words and bigrams, and body words and bigrams.

Method 16 is chosen to validate whether there is an on-line benefit for the popularity or Bayesian adjustments in Methods 6 and 7 by running the TF algorithm, with the same parameterization, unadjusted.

In Method 17, we investigate the impact of the recency-biased selection method by running Bayesian-adjusted TF but selecting the top-scoring articles regardless of age.

In Methods 18 and 19 we investigate the benefit of including the "negative interest filter." Recall that in other methods (with the exception of Method 2) if the user accepted two recommendations that were listed below a given recommended article, we inferred that the user was not interested in that article and refrained from recommending it in the future.

Finally, in Method 20, we make an adjustment to Method 7. In all of the other Bayesian-adjusted methods, when we figure the conditional $\Pr[s|(\text{not})\text{interested}]$, we use a *probability density function* (PDF) to determine the probability of getting precisely that score. In Method 20, by contrast, we use a *cumulative density function* (CDF) to determine the probability of getting a score at least that high.

3 Live Trial

In Phase II we received information about user visits in real time and had the opportunity to recommend articles that the user might be interested in and learn when our recommendations were taken. We were interested in answering two questions:

Question 1: Do any of the experimental methods described in Sect. 2.3 represent a significant improvement over the baseline methods in terms of the click-through rate (CTR)? That is, are users significantly more likely to accept recommendations made by certain methods than others?

Question 2: Do good recommendations increase the amount of time a user spends on the site? That is, do users who take recommendations have longer session lengths than users who do not, and is there a significant difference in user behavior after the user begins to take recommendations?

Click-through rate was chosen over metrics such as accuracy, precision, recall, or F-measure because in a live trial ground truth is unavailable for recommendations not taken and because users' preferences may change over time, so a recommendation skipped and later taken may still have been a mistake.

3.1 Experiment Protocol

For the live trial, Forbes Media identified a subset of visitors to the Forbes.com website whom they could stabily identify and made two changes to their served pages. First, whenever one of these users visited a page representing an article or slide deck, JavaScript code within the page made an asynchronous (Ajax) call to an HP Labs server passing in the URL of the visited page and an opaque numeric identifier representing the user. (The call requested an image, which was not displayed but which directed users worried about privacy concerns to a page explaining the experiment.) Second, whenever one of these users visited the Forbes.com homepage, the HP Labs server was requested to populate an HTML *iframe* with five recommended articles for the user identified in the request.

When a visit notification was received, the server first determined whether the associated user was already known and if not, selected a recommendation method to be used for them. Next, if the associated URL was not tied to a known article, the server requested the web page from the Forbes.com server and used it to extract features, including calling the concept extractor described in Sect. 2.2. If an analysis of the HTML code for the retrieved page indicated that it was part of a multi-page article, the server determined the entire "constellation" of URLs associated with the article and based the concept extraction on the text from all of the HTML pages.

The server then informed all of the recommendation methods about the visit, allowing them to update their models. Note that even though only one method would be used to make recommendations for a given user, other methods might also want to make use of the information. For example, in order to ensure that the score distributions used by the Bayesian adjustment were large enough, the positive distributions made use of visits from users assigned to any method, not merely those assigned to Bayesian-adjusted methods. Similarly, collaborative filtering methods made use of all visits, not merely visits by users assigned to collaborative filtering methods. It should be noted that in order to not impact the Forbes.com users, the actual visit notification was replied to immediately, with the described processing taking place as soon as possible after the fact.

When a recommendation iframe request was received, the server identified the recommendation method associated with the user (selecting one if necessary and associating it with the user for the future) and asked that method for five ordered articles to recommend. As noted above in Sect. 2.3, each recommendation method consisted of a scoring function, a set of filters, and a selection method. First, the current set of recommendable articles was passed through the filters, which typically did things like removing candidate articles that the user had already read. Next, the remaining candidates were passed to the scoring function, which, often using a constructed model for the user, computed a numeric score for each of them (possibly indicating for some of them that no score can be computed). Finally the selection method took the candidates and associated scores and picked a ranked list of five articles to recommend.

The list was then formatted as an HTML iframe containing links to the recommended articles, identified by their titles. The URLs of the articles were modified by the addition of an HTTP query parameter so that if the user clicked on one of them, the resulting visit notification received by the server would be seen to be an accepted recommendation, with the particular recommendation set and the rank within the list identified. The recommendation set itself, along with the user identifier, recommendation method, and timestamp, were stored on the server for later analysis.

The set of recommendable articles was taken to be those mentioned on a set of approximately 85 RSS feeds identified by Forbes Media, filtered to exclude those pages for which we would not receive a visit notification if a recommendation was taken. As with visit notifications, if the server did not already know about an article, it was fetched from Forbes.com, its page constellation determined, and its features extracted.

The system began receiving visit notifications on Sept. 29, 2011, and began making recommendations on Nov. 29, 2011. Phase II ended on Mar. 11, 2012. Excluding data associated with users involved in running the experiment and those whose behavior made them appear to be robots, we received 2.1 million visit notifications from 82,412 unique users (18% had a single visit). We made 388,392 recommendations, of which 3,118 (0.80%) were taken overall.

3.2 Comparison of Recommenders

As most non-trivial recommendation methods require a model of the user, and as building such a model typically requires observing user behavior, we are primarily concerned with the relative performance of different methods for recommendations given when a minimum number of observations have been made. For purposes of evaluation, we set our threshold at having observed the user visit at least five distinct articles. With this threshold, the system provided 311,015 recommendation lists, with a median of 13,789 recommendations given per recommendation method ($\max = 25,022$, $\min = 8,416$). Of these, 2,060 resulted in a recommendation being taken for an overall click-through rate of 0.66%. The click-through rates for all methods can be seen in Fig. 3.2. (Refer to Table 1 and Sect. 2.3 for descriptions of the various recommendation methods.)

Method 1, decayed popularity, represented the best of the baseline methods and was significantly ($p < 0.01$) better than any other baseline method, achieving a click-through rate of 1.02%. Three of the experimental methods exceeded this level—Method 6 (popularity-adjusted TF, at 1.14%), Method 7 (Bayesian-adjusted TF, at 1.11%), and Method 11 (Bayesian-adjusted TF/CF hybrid, at 1.40%)—but the first two comparisons are not significant ($p = 0.16$ and 0.24).

Method 11, the Bayesian-adjusted hybrid of content-based (TF) and collaborative filtering (CF) methods, however, was significantly better than decayed popularity and, indeed, all other methods ($p < 0.05$ vs. Methods 6 and 7 and $p < 0.01$ vs. all other methods). It achieved a click-through rate of 1.40%, which is a 37% improvement over decayed popularity and a 23% improvement over the next best method.

A full table of the significance of pairwise comparisons can be found in Table 2. Of particular interest is the relatively poor performance of using simple

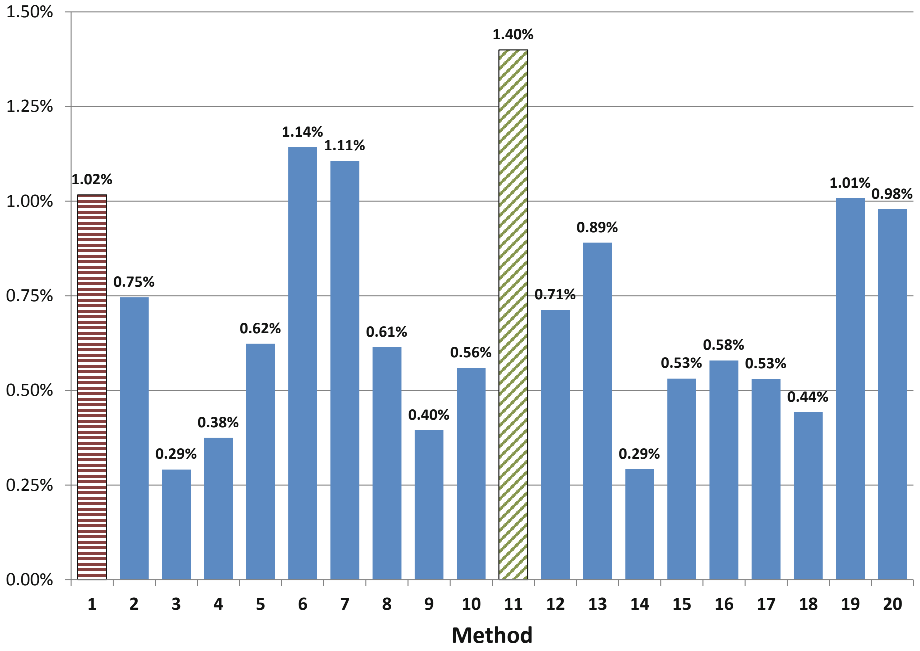


Fig. 1. Click-through rates by method for recommendations given with at least five articles viewed. See Sect. 2.3 and Table 1 for method descriptions.

undecayed popularity (Method 3), TF·IDF with plain "bag-of-words" features (Method 4), and simple Item-Item collaborative filtering (Method 5)—although between these "pure" methods, collaborative filtering was significantly ($p < 0.01$) better than TF·IDF.

Lesion Comparisons. Turning to the lesion methods described in Sect. 2.3, nearly all of them performed significantly worse than the experimental methods they were compared to.

The content-based methods that did not include Wikipedia concepts as features (Methods 12 and 13) performed significantly worse than the corresponding methods that did ($p < 0.01$ for popularity adjustment and $p < 0.05$ for Bayesian adjustment). The content-based methods that further did not include anchor phrases as features (Methods 14 and 15) performed significantly worse ($p < 0.01$) than either those with the full complement of features or those lacking concept features.

The adjustments themselves were helpful, as unadjusted TF (Method 16) performed significantly worse ($p < 0.01$) than either popularity-adjusted TF (Method 6) or Bayesian-adjusted TF (Method 7).

The recency focus for Bayesian-adjusted methods was useful, as the method that used it (Method 7) performed significantly better ($p < 0.01$) than the one that did not (Method 17).

Table 2. Significance levels of click-through rate comparisons by method for recommendations given with at least five articles viewed, ordered by click-through rate of method. See Sect. 2.3 and Table 1 for method descriptions.

🟩 Significant at P < 0.01
🟨 Significant at P < 0.05

Superior Method

Inferior Method	11	6	7	1	19	20	13	2	12	5	8	16	10	15	17	18	9	4	14
6	🟨																		
7	🟨	0.40																	
1	🟩	0.16	0.24																
19	🟩	0.17	0.25	0.47															
20	🟩	0.11	0.18	0.38	0.41														
13	🟩	🟨	🟨	0.12	0.17	0.22													
2	🟩	🟩	🟩	🟩	🟨	🟨	0.07												
12	🟩	🟩	🟩	🟩	🟨	🟨	0.07	0.38											
5	🟩	🟩	🟩	🟩	🟩	🟩	🟩	0.08	0.20										
8	🟩	🟩	🟩	🟩	🟩	🟩	🟩	0.08	0.19	0.46									
16	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	0.10	0.29	0.34								
10	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	0.08	0.23	0.28	0.41							
15	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	0.11	0.15	0.26	0.36						
17	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	0.15	0.19	0.29	0.38	0.50					
18	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	0.07	0.09	0.13				
9	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	0.22			
4	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	0.19	0.39		
14	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	0.11	
3	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	0.10	0.49

The negative interest filter, in which articles are suppressed that the user has bypassed twice in selecting a recommendation lower on the list, was a mixed bag. With popularity adjustment, the method that used it (Method 6) performed significantly better ($p < 0.01$) than the method that did not (Method 18). But with Bayesian adjustment, while the method that used it (Method 7) outperformed the method that did not (Method 19), the difference was not significant ($p = 0.25$). Finally, concerning the Bayesian adjustment, using PDFs (Method 7) was insignificantly better ($p = 0.18$) than using CDFs (Method 20).

3.3 Session Lengths

The second question we wanted to address was whether good recommendations translated into more page views. To measure this, we put all of the users into the same group, ignoring differences in their assigned recommendation methods, reasoning that regardless of method, a recommendation can be considered good if the user clicks on it.

To measure the number of page views, we broke each user's history into a number of *sessions*, with a session considered to have ended if more than sixty minutes elapsed between successive clicks. For each session, we noted whether at least one of the clicks was the result of taking a recommendation.

We found that sessions in which a recommendation was taken averaged 7.58 ± 0.35 clicks ($n = 2,750$), while sessions in which a recommendation was not taken averaged 3.56 ± 0.14 clicks ($n = 319,024$), for an increase of 4.02 clicks per session on average. Many of the sessions were very short, as users often visited the site by following a link, read an article, and left, never having been shown a recommendation. If such "trivial" sessions are excluded by setting a minimum session length of 3 clicks, sessions in which a recommendation was taken averaged 10.72 ± 0.51 clicks ($n = 1,822$), while sessions in which no recommendation was taken averaged 8.08 ± 0.42 ($n = 106,221$), for an increase of 2.30 clicks on average.

A reasonable question to ask is whether this difference is simply due to users who take recommendations being more likely to have longer sessions. This tendency does, indeed, exist. Sessions for users who ever took recommendations averaged 1.99 clicks longer (5.22 ± 0.05 as opposed to 3.24 ± 0.17) than sessions for those who never did. Indeed, even before the experiment proper began, when we were simply observing visits and not making recommendations, sessions for users who would eventually go on to take recommendations averaged 0.95 clicks longer (3.82 ± 0.08 as opposed to 2.87 ± 0.05). But this is unlikely to be the whole story, as when focusing on users who ever took clicks, their sessions averaged 4.45 ± 0.06 clicks before they took their first recommendation and 5.69 ± 0.08 clicks afterwards, for an increase of 1.24 clicks per session. For non-trivial sessions, the increase was greater, from 8.29 ± 0.11 clicks per session to 10.26 ± 0.15, for an average increase of 1.97 clicks per session.

All comparisons in this section are significant at the $p < 0.01$ level.

4 Discussion

The click-through rates presented in Sect. 3.2 are almost certainly underestimates. When our system was asked to make a recommendation, it had no information about articles that were displayed elsewhere on the page, and thus often recommended articles that were already among those selected by Forbes editors to appear on their page. When this happened, it is quite likely that users who agreed with our recommendations would choose the editor's link, which was larger and accompanied by a picture, unlike our title-only recommendations. Our system conservatively treats this situation as a miss.

Our experiments confirmed that sophisticated methods can (and, indeed, may be required to) beat simple popularity-based recommendation approaches. At a broad level, this collaboration demonstrates that, by partnering with a popular commercial website, it is possible to run meaningful large-scale experiments. Probably the biggest lesson we learned is that for experiments like these, up-front planning is key. This is drilled into medical researchers, but is somewhat foreign to most computer scientists. We are very grateful to Forbes Media for trusting us with their customers and their image. Being live on a commercial site meant that we had to be enormously careful that nothing that we did would cause them harm or embarrassment. Also, the scale of the experiment, particularly the amount of time that would be required to gather data to establish significance meant that we had to be prepared to turn it on and walk away for months, with only minor maintenance. This meant that we had to be confident that our code could handle the scale, but it also meant that we had to decide ahead of time what things we were going to want to compare and engineer the system so that we could test them all based on a single period of data collection. This meant that we spent more time than was normal before we began, but the delays were worthwhile.

5 Related Work

For a broad survey of recommendation methods see [1,23]. Our finding that hybrid methods excel is consistent with a broad literature. The Netflix competition enabled a large scale search of methods, of which the superior methods were consistently ensembles of many models, and the work turned toward optimization of these mixtures, e.g. [10]. Further, our finding that features from Wikipedia-based concepts [12] and a category hierarchy are consistent with others who have used taxonomies to attack the data sparseness, e.g. [11,14,18].

The vast majority of research publications evaluate methods using only historical datasets. While this may work for explicit user ratings of movies or books, passively collected datasets of user clicks on unrated news or other short-lived content may not represent user preference as much as which specific links were available or promoted on the web site at that time. A study of Digg popularity found this to be a very strong effect [16], and we have seen some confirming anecdotes in our datasets. Thus, we believe live trials are essential in this domain. Published studies using live feedback are relatively rare and often involve only small sets of users (e.g. 44 users in [3], 50 in [2], 57 in [18], 141 in [6], 239 in [20]). These sizes may support comparisons of content-analysis methods, but they give a large disadvantage to collaborative-filtering methods that depend on learning from a large user population. Further, such studies sometimes include in-depth questionnaires of perceptions and explicit rating by volunteers in a setting that does not match the intended live environment. Two studies of live news recommendation for large user populations (>10,000 users) were reported at Google. Their live trials compared relatively few methods: Liu et al. [17] reported testing a single method against their existing baseline in an A-B comparison; Das et al. [5] compared three algorithms, and also preceded their live trial with a broader search using their historical data.

One barrier to conducting such live studies is that researchers of recommendation methods are typically not situated within media companies. Our study shows that this bridge can be crossed with relatively little complication. A novel workaround proposed recently is to hire a user base via Mechanical Turk [21,13]. While this may be the best approach for some studies, it will always differ substantially from the intended production setting and its typical users.

A great deal of thoughtful work has been published on ways to evaluate recommendation methods, e.g. [7,8], including aspects of accuracy, novelty, diversity, explainability, and avoiding embarrassment. Koren [15] demonstrates that although most papers focus on error measures, esp. root-mean-squared-error (RMSE) for compatibility with the Netflix competition and its many papers, methods that optimize RMSE do not correspond to those that optimize the hit rate in the top-N list of recommendations—the common use-case for production. Hurley and Zhang [9] demonstrate that some increased emphasis on diversity can improve top-N recommendation. Online studies in recommendation and advertisement (another form of recommendation) usually measure by CTR, which aligns with financial incentives and implicitly factors in accuracy, novelty, diversity, etc., according to the preferences of the distribution of users.

6 Conclusions and Future Work

From the results of this substantial real-world experiment we can see that it is possible for a recommender system to outperform recommending recently popular articles and to do so not merely statistically significantly but also meaningfully, as our winning method had a click-through rate that represented a 37% improvement over the best popularity-based method. On the other hand, doing so is not easy. Our winning method made use of both content-based reasoning and collaborative filtering, and it required sophisticated analysis of the article text in order to provide the Wikipedia-based concept features it needed. Moreover, our lesion studies demonstrated that all of these attributes appear to be required.

We also saw in Sect. 3.3 that providing good recommendations can be useful from a business standpoint, as users who take recommendations stay on the site longer, which likely translates to more loyal customers and potentially greater advertising revenue.

In future experiments, it would be interesting to add a feedback mechanism that would allow the users to directly indicate when they found a listed recommendation to be poor and what they felt about an article (whether or not they viewed it via a recommendation link). In the current work, we could only infer that the user found an article interesting if and only if they viewed it, but there's always the possibility that the user was "tricked" into clicking on the headline of an article they actually were not interested in. With direct feedback, the recommendation methods would be expected to be more accurate and more dynamic, and users might well find recommendations more appealing if they felt that they had a measure of control.

In the same vein, future experiments would profit from being told what other articles were referred to on pages. This would allow the system to more accurately detect lack of interest in articles or that one article is less interesting than another, and it would allow the recommender to avoid duplicating articles that are already displayed on the page.

In the current experiment, the only thing the recommenders knew about the users came from the pages they viewed on the site. In future experiments, it could be useful to add other sources of information, e.g., demographics or other behavior known by the site.

Finally, in this experiment recommendations were produced only on the site's main page. This meant that unless the user navigated back to that page rather than taking a link from the article page, the system had no chance to make use of the article being viewed to immediately display options for where to go next. Indeed, the system may never get the chance to make a recommendation to users that arrive at the site from inward-pointing links and do not use the main page. In future experiments, it would be helpful to be allowed to make recommendations on article pages as well.

Acknowledgments. We would like to thank David Dunlop, Carol Ozaki, Craig Sayers, Sherry Xia, and the staff at Forbes who helped this project be a success.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems. IEEE Trans. on Knowl. and Data Eng. 17(6), 734–749 (2005)
2. Billsus, D., Pazzani, M.J.: Adaptive News Access. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 550–570. Springer, Heidelberg (2007)
3. Chen, J., Nairn, R., Nelson, L., Bernstein, M., Chi, E.: Short and tweet: experiments on recommending content from information streams. In: CHI 2010 (2010)
4. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: RecSys 2010, pp. 39–46 (2010)
5. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: WWW 2007, pp. 271–280 (2007)
6. de Wit, J.: Evaluating recommender systems: an evaluation framework to predict user satisfaction for recommender systems in an electronic programme guide context. Master's thesis, University of Twente, The Netherlands (May 2008)
7. Ge, M., Delgado-Battenfeld, C., Jannach, D.: Beyond accuracy: evaluating recommender systems by coverage and serendipity. In: RecSys 2010, p. 257 (2010)
8. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 22(1), 5–53 (2004)
9. Hurley, N., Zhang, M.: Novelty and diversity in top-N recommendation – analysis and evaluation. ACM Trans. Internet Technol. 14, 14:1–14:30 (2011)
10. Jahrer, M., Töscher, A., Legenstein, R.: Combining predictions for accurate recommender systems. In: KDD 2010, pp. 693–702 (2010)
11. Katz, G., Ofek, N., Shapira, B., Rokach, L., Shani, G.: Using Wikipedia to boost collaborative filtering techniques. In: RecSys 2011, pp. 285–288 (2011)
12. Kirshenbaum, E.: A Wikipedia-based concept extractor for unstructured text. Technical report, HP Labs (in preparation)
13. Kittur, A., Chi, E.H., Suh, B.: Crowdsourcing user studies with mechanical turk. In: CHI 2008, pp. 453–456 (2008)
14. Koenigstein, N., Dror, G., Koren, Y.: Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In: RecSys 2011 (2011)
15. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD 2008, pp. 426–434 (2008)
16. Lerman, K., Hogg, T.: Using a model of social dynamics to predict popularity of news. In: WWW 2010, pp. 621–630 (2010)
17. Liu, J., Dolan, P., Pedersen, E.R.: Personalized news recommendation based on click behavior. In: IUI 2010, pp. 31–40 (2010)
18. Maidel, V., Shoval, P., Shapira, B., Taieb-Maimon, M.: Evaluation of an ontology-content based filtering method for a personalized newspaper. In: RecSys 2008 (2008)
19. Pan, R., Zhou, Y., Cao, B., Liu, N.N., Lukose, R., Scholz, M., Yang, Q.: One-class collaborative filtering. In: ICDM 2008, pp. 502–511 (2008)
20. Pu, P., Chen, L., Hu, R.: A user-centric evaluation framework for recommender systems. In: RecSys 2011, pp. 157–164 (2011)
21. Sandholm, T., Ung, H., Aperjis, C., Huberman, B.A.: Global budgets for local recommendations. In: RecSys 2010, pp. 13–20 (2010)
22. Steck, H.: Item popularity and recommendation accuracy. In: RecSys 2011 (2011)
23. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. In: Advances in Artificial Intelligence, pp. 4:2–4:2 (January 2009)
24. Zheng, H., Wang, D., Zhang, Q., Li, H., Yang, T.: Do clicks measure recommendation relevancy? an empirical user study. In: RecSys 2010, pp. 249–252 (2010)

Fast ALS-Based Tensor Factorization for Context-Aware Recommendation from Implicit Feedback

Balázs Hidasi[1,2,⋆] and Domonkos Tikk[1]

[1] Gravity R&D Ltd.
[2] Budapest University of Technology and Economics
{balazs.hidasi,domonkos.tikk}@gravityrd.com

Abstract. Albeit the implicit feedback based recommendation problem—when only the user history is available but there are no ratings—is the most typical setting in real-world applications, it is much less researched than the explicit feedback case. State-of-the-art algorithms that are efficient on the explicit case cannot be straightforwardly transformed to the implicit case if scalability should be maintained. There are few implicit feedback benchmark datasets, therefore new ideas are usually experimented on explicit benchmarks. In this paper, we propose a generic context-aware implicit feedback recommender algorithm, coined iTALS. iTALS applies a fast, ALS-based tensor factorization learning method that scales linearly with the number of non-zero elements in the tensor. The method also allows us to incorporate various contextual information into the model while maintaining its computational efficiency. We present two context-aware implementation variants of iTALS. The first incorporates seasonality and enables to distinguish user behavior in different time intervals. The other views the user history as sequential information and has the ability to recognize usage pattern typical to certain group of items, e.g. to automatically tell apart product types that are typically purchased repetitively or once. Experiments performed on five implicit datasets (LastFM 1K, Grocery, VoD, and "implicitized" Netflix and MovieLens 10M) show that by integrating context-aware information with our factorization framework into the state-of-the-art implicit recommender algorithm the recommendation quality improves significantly.

Keywords: recommender systems, tensor factorization, context awareness, implicit feedback.

1 Introduction

Recommender systems are information filtering algorithms that help users in information overload to find interesting items (products, content, etc). Users get personalized recommendations that contain typically a few items deemed to be of user's interest. The relevance of an item with respect to a user is predicted by

[⋆] B. Hidasi was supported by TÁMOP-4.2.2.B-10/1–2010-0009.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 67–82, 2012.

recommender algorithms; items with the highest prediction scores are displayed to the user.

Recommender algorithms are usually sorted into two main approaches: the content based filtering (CBF) and the collaborative filtering (CF). CBF algorithms use user metadata (e.g. demographic data) and item metadata (e.g. author, genre, etc.) and predict user preference using these attributes. In contrast, CF methods do not use metadata, but only data of user–item interactions. Depending on the nature of the interactions, CF algorithms can be further classified into explicit and implicit feedback based methods. In the former case, users provide explicit information on their item preferences, typically in form of user ratings. In the latter case, however, users express their item preferences only implicitly, as they regularly use an online system; typical implicit feedback types are viewing and purchasing. Obviously, implicit feedback data is less reliable as we will detail later. CF algorithms proved to be more accurate than CBF methods, if sufficient preference data is available [1].

CF algorithms can be classified into memory-based and model-based ones. Until recently, memory-based solutions were concerned as the state-of-the-art. These are neighbor methods that make use of item or user rating vectors to define similarity, and they calculate recommendations as a weighted average of similar item or user rating vectors. In the last few years, model-based methods gained enhanced popularity, because they were found to be much more accurate in the Netflix Prize, a community contest launched in late 2006 that provided the largest explicit benchmark dataset (100M ratings) [2] for a long time.

Model-based methods build generalized models that intend to capture user preference. The most successful approaches are the latent factor algorithms. These represent each user and item as a feature vector and the rating of user u for item i is predicted as the scalar product of these vectors. Different matrix factorization (MF) methods are applied to compute these vectors, which approximate the partially known rating matrix using alternating least squares (ALS) [3], gradient [4] and coordinate descent method [5], conjugate gradient method [6], singular value decomposition [7], or a probabilistic framework [8].

Explicit feedback based methods are able to provide accurate recommendations if enough ratings are available. In certain application areas, such as movie rental, travel applications, video streaming, users have motivation to provide ratings to get better service, better recommendations, or award or punish a certain vendor. However, in general, users of an arbitrary online service do not tend to provide ratings on items even if such an option is available, because (1) when purchasing they have no information on their satisfaction (2) they are not motivated to return later to the system to rate. In such cases, user preferences can only be inferred by interpreting user actions (also called *events*). For instance, a recommender system may consider the navigation to a particular product page as an implicit sign of preference for the item shown on that page [9]. The user history specific to items are thus considered as implicit feedback on user taste. Note that the interpretation of implicit feedback data may not necessarily reflect user satisfaction which makes the implicit feedback based preference modeling a difficult task. For instance, a purchased item could be disappointing for the user, so it might not mean a positive feedback. We can neither interpret missing navigational or purchase information as negative feedback, that is, such information is not available.

Despite its practical importance, this harder but more realistic task has been less studied. The proposed solutions for the implicit task are often the algorithms for the explicit problems that had been modified in a way that they can handle the implicit task.

The classical MF methods only consider user-item interaction (ratings or events) when building the model. However, we may have additional information related to items, users or events, which are together termed *contextual information*, or briefly *context*. Context can be, for instance, the time or location of recommendation, social networks of users, or user/item metadata [10]. Integrating context can help to improve recommender models. Tensor factorization have been suggested as a generalization of MF for considering contextual information [11]. However, the existing methods only work for the explicit problem. In this work, we developed a tensor factorization algorithm that can efficiently handle the implicit recommendation task.

The novelty of our work is threefold: (1) we developed a fast tensor factorization method—coined iTALS—that can efficiently factorize huge tensors; (2) we adapted this general tensor factorization to the implicit recommendation task; (3) we present two specific implementations of this general implicit tensor factorization that consider different contextual information. The first variant uses seasonality which was also used in [11] for the explicit problem. The second algorithm applies sequentiality of user actions and is able to learn association rule like usage patterns. By using these patterns we can tell apart items or item categories having been purchased with different repetitiveness, which improves the accuracy of recommendations. To our best knowledge, iTALS is the first factorization algorithm that uses this type of information.

This paper is organized as follows. Section 2 briefly reviews related work on context-aware recommendation algorithms and tensor factorization. In Section 3 we introduce our tensor factorization method and its application to the implicit recommendation task. Section 4 shows two application examples of our factorization method: (1) we show how seasonality can be included in recommendations and (2) we discuss how a recommendation algorithm can learn repetitiveness patterns from the dataset. Section 5 presents the results of our experiments, and Section 6 sums up our work and derive the conclusions.

1.1 Notation

We will use the following notation in the rest of this paper:

- $A \circ B \circ \ldots \rightarrow$ The Hadamard (elementwise) product of A, B, ... The operands are of equal size, and the result's size is also the same. The element of the result at $(i, j, k, \ldots)$ is the product of the element of A, B, ... at $(i, j, k, \ldots)$. This operator has higher precedence than matrix multiplication in our discussion.

- $A_{\bullet,i}/A_{i,\bullet}$ → The i^{th} column/row of matrix A.
- $A_{i_1,i_2,\dots}$ → The $(i_1, i_2, \dots)$ element of tensor/matrix A.
- K → The number of features, the main parameter of factorization.
- D → The number of dimensions of the tensor.
- T → A D dimensional tensor that contains only zeroes and ones (preference tensor).
- W → A tensor with the exact same size as T (weight tensor).
- S_i → The size of T in the i^{th} dimension $(i = 1, \dots, D)$.
- N^+ → The number of non-zero elements in tensor T.
- $M^{(i)}$ → A $K \times S_i$ sized matrix. Its columns are the feature vectors for the entities in the i^{th} dimension.

2 Related Work

Context-aware recommender systems [12] emerged as an important research topic in the last years and entire workshops are devoted to this topic on major conferences (CARS series started in 2009 [13], CAMRA in 2010 [14]). The application fields of context-aware recommenders include among other movie [15] and music recommendation [16], point-of-interest recommendation (POI) [17], citation recommendation [18]. Context-aware recommender approaches can be classified into three main groups: pre-filtering, post-filtering and contextual modeling [10]. Baltrunas and Amatriain [16] proposed a pre-filtering approach by partitioned user profiles into *micro-profiles* based on the time split of user event falls, and experimented with different time partitioning. Post-filtering ignores the contextual data at recommendation generation, but disregards irrelevant items (in a given context) or adjust recommendation score (according to the context) when the recommendation list is prepared; see a comparison in [19]. The tensor factorization based solutions, including our proposed approach, falls into the contextual modeling category.

Tensor factorization incorporates contextual information into the recommendation model. Let us have a set of items, users and ratings (or events) and assume that additional context of the ratings is available (e.g. time of the rating). Having C different contexts, the rating data can be cast into a $C + 2$ dimensional tensor. The first dimension corresponds to users, the second to items and the subsequent C dimensions $[3, \dots, C + 2]$ are devoted to contexts. We want to decompose this tensor into lower rank matrices and/or tensors in a way that the reconstruction the original tensor from its decomposition approximates well the original tensor. Approximation accuracy is calculated at the known positions of the tensor using RMSE as error measure. In [11], a sparse HOSVD [20] method is presented that decomposes a D dimensional sparse tensor into D matrices and a D dimensional tensor. If the size of the original tensor is $S_1 \times S_2 \times \cdots \times S_D$ and the number of features is K then the size of the matrices are $S_1 \times K$, $S_2 \times K$, ..., $S_D \times K$ and the size of the tensor is $K \times K \times \cdots \times K$. The authors use gradient descent on the known ratings to find the decomposition, and by doing so, the complexity of one iteration of their algorithm scales *linearly* with the number of non-missing values in the original tensor (number of rating) and *cubically* with the number of features (K). This is much less than the cost of the dense HOSVD, which is $O(K\cdot(S_1+\cdots+S_D)^D)$. A further improvement was proposed by Rendle *et al* [21], where the computational complexity was reduced so that their method scales linearly *both* with the number of explicit ratings and with the number of features. However, if the original tensor is large and dense like for the implicit recommendation task then neither method scales well.

3 ALS Based Fast Tensor Factorization

In this section we present iTALS, a general ALS-based tensor factorization algorithm that scales linearly with the non-zero element of a dense tensor (when appropriate weighting is used) and cubically with the number of features. This property makes our algorithm suitable to handle the context-aware implicit recommendation problem.

Let T be a tensor of zeroes and ones and let W contain weights to each element of T. $T_{u,i,c_1,\cdots,c_C}$ is 1 if user u has (at least one) event on item i while the context-state of j^{th} context dimension was c_j, thus the proportion of ones in the tensor is very low. An element of W is 1 if the corresponding element in T is 0 and greater than 1 otherwise. Instead of using the form of the common HOSVD decomposition (D matrices and a D dimensional tensor) we decompose the original T tensor into D matrices. The size of the matrices are $K\times S_1, K\times S_2,\dots,K\times S_D$. The prediction for a given cell in T is the elementwise product of columns from $M^{(i)}$ low rank matrices. Equation 1 describes the model.

$$\hat{T}_{i_1,i_2,\dots,i_D}=1^T M^{(1)}_{\bullet,i_1}\circ M^{(2)}_{\bullet,i_2}\circ\cdots\circ M^{(D)}_{\bullet,i_D} \tag{1}$$

We want to minimize the loss function of equation 2:

$$L(M^{(1)},\dots,M^{(D)})=\sum_{i_1=1,\dots,i_D=1}^{S_1,\dots,S_D} W_{i_1,\dots,i_D}\left(T_{i_1,\dots,i_D}-\hat{T}_{i_1,\dots,i_D}\right)^2 \tag{2}$$

If all but one $M^{(i)}$ is fixed, L is convex in the non-fixed variables. We use this method to minimize the loss function. L reaches its minimum (in $M^{(i)}$) where its derivate with respect to $M^{(i)}$ is zero. Since the derivate of L is linear in $M^{(i)}$ the columns of the matrix can be computed separately. For the $(i_1)^{\text{th}}$ column of $M^{(1)}$:

$$\begin{gathered}0=\frac{\partial L}{\partial M^{(1)}_{\bullet,i_1}}=-2\underbrace{\sum_{i_2=1,\dots,i_D=1}^{S_2,\dots,S_D} W_{i_2,\dots,i_D}T_{i_1,\dots,i_D}\left(M^{(2)}_{\bullet,i_2}\circ\cdots\circ M^{(D)}_{\bullet,i_D}\right)}_{\mathcal{O}}+\\ \underbrace{2\sum_{i_2=1,\dots,i_D=1}^{S_2,\dots,S_D} W_{i_2,\dots,i_D}\left(M^{(2)}_{\bullet,i_2}\circ\cdots\circ M^{(D)}_{\bullet,i_D}\right)\left(M^{(2)}_{\bullet,i_2}\circ\cdots\circ M^{(D)}_{\bullet,i_D}\right)^T M^{(1)}_{\bullet,i_1}}_{\mathcal{I}}\end{gathered} \tag{3}$$

It takes $O(DKN_{i_1}^+)$ time to compute $\mathcal{O}$ in equation 3, because only $N_{i_1}^+$ cells of T for i_1 in the first dimension contain ones, the others are zeroes. For every column it yields a complexity of $O(DKN^+)$. The naive computation of $\mathcal{I}$ however is very expensive computationally: $O(K\prod_{i=2}^D S_i)$. Therefore we transform $\mathcal{I}$ by using $W_{i_2,\dots,i_D} = W'_{i_2,\dots,i_D} + 1$ and get:

$$\begin{aligned}\mathcal{I} = &\sum_{i_2=1,\dots,i_D=1}^{S_2,\dots,S_D} W'_{i_2,\dots,i_D}\left(M_{\bullet,i_2}^{(2)}\circ\dots\circ M_{\bullet,i_D}^{(D)}\right)\left(M_{\bullet,i_2}^{(2)}\circ\dots\circ M_{\bullet,i_D}^{(D)}\right)^T M_{\bullet,i_1}^{(1)} + \\ &+ \underbrace{\sum_{i_2=1,\dots,i_D=1}^{S_2,\dots S_D}\left(M_{\bullet,i_2}^{(2)}\circ\dots\circ M_{\bullet,i_D}^{(D)}\right)\left(M_{\bullet,i_2}^{(2)}\circ\dots\circ M_{\bullet,i_D}^{(D)}\right)^T M_{\bullet,i_1}^{(1)}}_{\mathcal{J}}\end{aligned} \tag{4}$$

The first part in equation 4 can be calculated in $O(K^2N_{i_1}^+)$ as $W'_{i_2,\dots,i_D} = (W_{i_2,\dots,i_D} - 1)$ and the weights for the zero elements of T are ones. This step is the generalization of the Hu *et. al*'s adaptation of ALS to the implicit problem [22]. The total complexity of calculating all columns of the matrix is $O(K^2N^+)$. $\mathcal{J}$ is the same for all columns of $M^{(1)}$ (independent of i_1) and thus can be precomputed. However the cost of directly computing $\mathcal{J}$ remains $O(K\prod_{i=2}^D S_i)$. Observe the following:

$$\begin{aligned}\mathcal{J}_{j,k} &= \left(\sum_{i_2=1,\dots,i_D=1}^{S_2,\dots,S_D}\left(M_{\bullet,i_2}^{(2)}\circ\dots\circ M_{\bullet,i_D}^{(D)}\right)\left(M_{\bullet,i_2}^{(2)}\circ\dots\circ M_{\bullet,i_D}^{(D)}\right)^T\right)_{j,k} = \\ &= \sum_{i_2=1,\dots,i_D=1}^{S_2,\dots,S_D}\left(M_{j,i_2}^{(2)}\cdot\dots\cdot M_{j,i_D}^{(D)}\right)\left(M_{k,i_2}^{(2)}\cdot\dots\cdot M_{k,i_D}^{(D)}\right) = \\ &= \left(\sum_{i_2=1}^{S_2} M_{j,i_2}^{(2)}M_{k,i_2}^{(2)}\right)\cdot\dots\cdot\left(\sum_{i_D=1}^{S_D} M_{j,i_D}^{(D)}M_{k,i_D}^{(D)}\right)\end{aligned} \tag{5}$$

Using equation 5 we can transform the second part from equation 4 into the following form:

$$\begin{aligned}\mathcal{J} &= \sum_{i_2=1,\dots,i_D=1}^{S_2,\dots,S_D}\left(M_{\bullet,i_2}^{(2)}\circ\dots\circ M_{\bullet,i_D}^{(D)}\right)\left(M_{\bullet,i_2}^{(2)}\circ\dots\circ M_{\bullet,i_D}^{(D)}\right)^T = \\ &= \underbrace{\left(\sum_{i_2=1}^{S_2} M_{\bullet,i_2}^{(2)}\left(M_{\bullet,i_2}^{(2)}\right)^T\right)}_{\mathcal{M}^{(2)}}\circ\dots\circ\underbrace{\left(\sum_{i_D=1}^{S_D} M_{\bullet,i_D}^{(D)}\left(M_{\bullet,i_D}^{(D)}\right)^T\right)}_{\mathcal{M}^{(D)}}\end{aligned} \tag{6}$$

The members of equation 6 can be computed in $O(S_iK^2)$ time. From the $\mathcal{M}^{(i)}$ matrices the expression can be calculated in $O(K^2D)$ time. Note that the $\mathcal{M}^{(i)}$ is needed for computing all but the i^{th} matrix but only changes if $M^{(i)}$ changed.

Therefore we count the cost of computing $\mathcal{M}^{(i)}$ to the cost of recomputing $M^{(i)}$. To get the desired column of the matrix we need to invert a $K \times K$ sized matrix per column (see equation 3). That requires $O(K^3 S_1)$ time for all columns of $M^{(1)}$. The columns of the other matrices can be calculated similarly.

Algorithm 1. Fast ALS-based tensor factorization for implicit feedback recommendations

Input: T: a D dimensional $S_1 \times \cdots \times S_D$ sized tensor of zeroes and ones; W: a D dimensional $S_1 \times \cdots \times S_D$ sized tensor containing the weights; K: number of features; E: number of epochs

Output: $\{M^{(i)}\}_{i=1,\dots,D}$ $K \times S_i$ sized low rank matrices

```
procedure iTALS($T$, $W$, $K$, $E$)
for $i = 1, \dots, D$ do
    $M^{(i)} \leftarrow$ Random $K \times S_i$ sized matrix
    $\mathcal{M}^{(i)} \leftarrow M^{(i)}(M^{(i)})^T$
end for
for $e = 1, \dots, E$ do
    for $i = 1, \dots, D$ do
        $C^{(i)} \leftarrow \mathcal{M}^{(\ell_1)} \circ \cdots \circ \mathcal{M}^{(\ell_{D-1})}, (i \notin \{\ell_1, \dots, \ell_{D-1}\})$
        $T^{(i)} \leftarrow$ UnfoldTensor($T$,$i$)
        for $j_i = 1, \dots, S_i$ do
            $C^{(i)}_{j_i} \leftarrow C^{(i)}$
            $O^{(i)}_{j_i} \leftarrow 0$
            for all $t : \{t \in T^{(i)}_{j_i}, t \neq 0\}$ do
                $\{j_\ell | \ell \neq i\} \leftarrow$ Indices of $t$ in $T$
                $W_t \leftarrow$ GetWeight($W$,$t$)
                $v \leftarrow M^{(\ell_1)} \circ \cdots \circ M^{(\ell_{D-1})}, (i \notin \{\ell_1, \dots, \ell_{D-1}\})$
                $C^{(i)}_{j_i} \leftarrow C^{(i)}_{j_i} + v W_t v^T$ and $O^{(i)}_{j_i} \leftarrow O^{(i)}_{j_i} + W_t v$
            end for
            $M^{(i)}_{\bullet,j_i} \leftarrow (C^{(i)}_{j_i} + \lambda I)^{-1} O^{(i)}_{j_i}$
        end for
        $\mathcal{M}^{(i)} \leftarrow M^{(i)}(M^{(i)})^T$
    end for
end for
return $\{M^{(i)}\}_{i=1 \dots D}$
end procedure
```

The total cost of computing $M^{(i)}$ is $O(K^3 S_i + K^2 N^+ + KDN^+)$ that can be simplified to $O(K^3 S_i + K^2 N^+)$ using that usually $D \ll K$. Therefore the cost of computing each matrix once is $O\left(K^3 \sum_{i=1}^{D} S_i + K^2 N^+\right)$. Thus the cost of an epoch is linear in the number of the non-zero examples and cubical in the number of features. The cost is also linear in the number of dimensions of the tensor and the sum of the length of the tensors in each dimension. We will also show in Section 5.1 that the $O(K^2)$ part is dominant when dealing with practical problems. The complexity of the gradient descent method for implicit feedback is $O(K \prod_{i=1}^{D} S_i)$ that is linear in the number of features but the $\prod_{i=1}^{D} S_i$ part makes impossible to run it on real life datasets. Sampling can be applied to reduce that cost but it is not trivial how to sample in the implicit feedback case.

The pseudocode of the suggested iTALS (Tensor factorization using ALS for implicit recommendation problem) is given in Algorithm 1. There we use two simple functions. UNFOLDTENSOR(T, i) unfolds tensor T by its i^{th} dimension. This step is used for the sake of clarity, but with proper indexing we would not need to actually unfold the tensor. GETWEIGHT(W, t) gets the weight from the weight tensor W for the t element of tensor T and creates a diagonal matrix from it. The size of W_t is $K \times K$ and it contains the weight for t in its main diagonal and 0 elsewhere. The pseudocode follows the deduction above. In line 3 we precompute $\mathcal{M}^{(i)}$. We create the column independent part from equation 4 in line 7. We add the column dependent parts to each side of equation 3 in lines 12–17 and compute the desired column in line 18. In this step we use regularization to avoid numerical instability and overfitting of the model. After each column of $M^{(i)}$ is computed $\mathcal{M}^{(i)}$ is recomputed in line 20.

4 Context-Aware iTALS Algorithm

In this section we derive two specific algorithms from the generic iTALS method presented in Section 3. The first method uses seasonality as context, the second considers the user history as sequential data, and learns meta-rules about sequentiality and repetitiveness.

4.1 Seasonality

Many application areas of recommender systems exhibit the seasonality effect, therefore seasonal data is an obvious choice as context [23]. Strong periodicity can be observed in most of the human activities: as people have regular daily routines, they also follow similar patterns in TV watching at different time of a day, they do their summer/winter vacation around the same time in each year. Taking the TV watching example, it is probable that horror movies are typically watched at night and animation is watched in the afternoon or weekend mornings. Seasonality can be also observed in grocery shopping or in hotel reservation data.

In order to consider seasonality, first we have to define the length of season. During a season we do not expect repetitions in the aggregated behavior of users, but we expect that at the same time offset in different seasons, the aggregated behavior of the users will be similar. The length of the season depends on the data. For example it is reasonable to set the season length to be 1 day for VoD consumption, however, this is not an appropriate choice for shopping data, where 1 week or 1 month is more justifiable. Having the length of the season determined, we need to create *time bands* (bins) in the seasons. These time bands are the possible context-states. Time bands specify the time resolution of a season, which is also data dependent. We can create time bands with equal or different length.

For example, every day of a week are time bands of equal length, but 'morning', 'around noon', 'afternoon', 'evening', 'late evening', 'night' could be time bands of a day with different length. Obviously, these two steps require some a-priori knowledge about the data or the recommendation problem, but iTALS is not too sensitive to minor deviations related to the length and the resolution of the season.

In the next step, events are assigned to time bands according to their time stamp. Thus, we can create the (user, item, time band) tensor. We factorize this tensor using the iTALS algorithm and we get feature vectors for each user, for each item and for each time band. When a recommendation is requested for user u at time t, first the time band of t is determined and then the preference value for each item using the feature vector of user u and the feature vector of time band tb_t is calculated.

4.2 Sequentiality

Recommendation algorithms often recommend items from categories that the user likes. For example if the user often watches horror movies then the algorithm will recommend her horror movies. This phenomenon is even stronger if time decay is applied and so recent events have greater weights. Pushing newer events can increase accuracy, because similar items will be recommended. This functioning can be beneficial in some application fields, like VoD recommendation, but will fail in such cases where repetitiveness in user behavior with respect to items can not be observed. A typical example for that is related to household appliance products: if a user buys a TV set and then she gets further TV sets recommended, she will not probably purchase another one. In such a case, complementary or related goods are more appropriate to recommend, DVD players or external TV-tuners for example. On the other hand, the purchase of a DVD movie does not exclude at all the purchase of another one. Whether recommendation of similar items is reasonable, depends on the nature of the item and behavior of the user. Next, we propose an approach to integrate the repetitiveness of purchase patterns into the latent factor model.

Using association rules is a possible approach to specify item purchase patterns. Association rules [24] are often used to determine which products are bought frequently together and it was reported that in certain cases association rule based recommendations yield the best performance [25]. In our setting, we can extract purchase patterns from the data using association rule mining on the subsequent user events within a given time window. There are two possibilities: we can generate category–category rules, or category–item rule, thus having usage patterns:

- if a user bought an item from category A then she will buy an item from category B next time, or
- if a user bought an item from category A then she will buy an item X next time.

We face, however, with the following problems, when attempting to use such patterns in recommendations: (1) the parameter selection (minimum support, minimum confidence and minimum lift) influences largely the performance, their optimization may be slow; (2) rules with negated consequents (e.g. bought from A will not buy from B) are not found at all; (3) with category–category rules one should devise further weighting/filtering to promote/demote the items in the pushed category; (4) the category–item rules are too specific therefore either one gets too many rules or the rules will overfit.

We show how repetitiveness related usage patterns can be efficiently integrated into recommendation model using the the iTALS algorithm. Let us now consider the *category of last purchased item* as the context for the next recommendation. The tensor has again three dimensions: users, items and item categories. The (i, u, c) element of the tensor means that user u bought item i and the user's latest purchase (before buying i) was an item from category c. Using the examples above: the user bought a given DVD player after the purchase of a TV set. After factorizing this tensor we get feature vectors for the item categories as well. These vectors act as weights in the feature space that reweight the user–item relations. For example, assuming that the first item feature means “having large screen” then the first feature of the TV category would be low as such items are demoted. If the second item feature means “item can play discs” then the second feature of the TV category would be high as these items are promoted.

The advantage of this method is that it learns the usage patterns from the data globally by producing feature vectors that reweight the user–item relations. One gets simple but general usage patterns using the proposed solution that integrates seamlessly into the common factorization framework: no post-processing is required to define promotional/demotional weights/filters.

We can generalize the concept described above to take into account several recent purchases. We could create a $C + 2$ dimensional tensor, where the $[3, \ldots, C + 2]$ dimensions would represent the item categories of the last C purchases, but the resulting tensor would be very sparse as we increase C. Instead we remain at a three dimensional tensor but we set simultaneously C item categories to 1 for each user–item pair. We may also decrease the weights in W for those additional $C - 1$ cells as they belong to older purchases. Thus we may control the effect of previous purchases based on their recency. When recommending, we have to compute the (weighted) average of the feature vectors of the corresponding categories and use that vector as the context feature vector.

5 Experiments

We used five databases to validate our algorithms. Three of them contain genuine implicit feedback data (LastFM 1K and 2 proprietary), while the other two are implicit variants of explicit feedback data. The *LastFM 1K* [26] dataset contains listening habits of ∼1 000 users on songs of ∼170 000 artists (artists are considered items). The training set contains all events until 28/04/2009. The test set contains the events of the next day following the training period. In *VoD*

Table 1. Recall@20 for all datasets and algorithms using factorization with 20 and 40 features; in each row, the best and second best results are highlighted by bold and slanted typesetting, respectively

Dataset	**iALS**	**iCA baseline**	**iTALS**	**iTALS**	**iTALS**	**iTALS**
		time bands	**time bands**	**seq.**	**seq. (2)**	**seq. (5)**
VOD (20)	0.0632	*0.0847*	**0.1125**	0.0689	0.0678	0.0666
VOD (40)	0.0753	*0.0910*	**0.1240**	0.0855	0.0930	0.0883
Grocery (20)	0.0656	0.0803	0.1032	**0.1261**	*0.1223*	0.1153
Grocery (40)	0.0707	0.0872	0.1081	*0.1340*	**0.1351**	0.1189
LastFM 1K (20)	0.0157	0.0249	0.0352	*0.0747*	**0.0793**	0.0733
LastFM 1K (40)	0.0333	0.0351	0.0418	0.0785	**0.0851**	*0.0800*
Netflix (20)	0.0540	*0.0593*	**0.0724**	0.0512	0.0534	0.0537
Netflix (40)	0.0552	*0.0561*	**0.0671**	0.0503	0.0527	0.0538
MovieLens (20)	0.0494	*0.0553*	**0.0896**	0.0406	0.0450	0.0457
MovieLens (40)	0.0535	0.0494	**0.0937**	0.0361	0.0480	*0.0498*

consumption dataset, with 8 weeks of training data we tested on the data of the next day. Thus, all test events occurred after the last train event. The training set contains 22.5 million events and 17 000 items. The online *grocery* dataset contains only purchase events. We used a few years' data for training and one month for testing. The training set contains 6.24 million events and 14 000 items. The two explicit feedback datasets are the Netflix [2] and the MovieLens 10M [27]. We kept the five star ratings for the former and ratings of 4.5 and above for the latter and used them as positive implicit feedback. For train-test splits we used the splitting dates 15/12/2005 and 01/12/2008, respectively.

We determined the seasonality for each dataset, that is, the periodicity patterns observed in the data. As for the VoD data, we defined a day as the season and defined custom time intervals as time bands ('morning', 'around noon', 'afternoon', 'evening', 'late evening', 'night' and 'dawn'), because people watch and channels broadcast different programs at different time of the day. For LastFM 1K and MovieLens we also used a day as the season and time bands of 30 minutes. For the Grocery data we defined a week as the season and the days of the week as the time bands. The argument here is that people tend to follow different shopping behavior on weekdays and weekends. For the Netflix data only the day of the rating is available, so we decided to define a week as the season and the days of the week as time bands.

In our next experiment, we used item category with Grocery and Netflix datasets, genre with VoD and MovieLens and artists for LastFM as the category of the item for the meta-rule learning algorithm. We experimented with using the last 1, 2, 5 events prior to the current event of the users.

We compared the two iTALS variants to the basic iALS as well as to a context-aware baseline for implicit feedback data. This method, referred as *implicit CA (iCA) baseline*, is the composite of several iALS models. For each context state we train a model using only the events with the appropriate context, e.g., with the VoD we train 7 models for the 7 time bands. The context of the recommendation request (e.g. time of day) selects the model for the prediction. This baseline treats context-states independently. Due to its long running time we used iCA only with seasonality, as #(time bands) $\ll$ #(preceding item categories).

Every algorithm has three common parameters: the number of features, the number of epochs and the regularization parameter. We set the number of features to 20 and 40 commonly used in literature [1,7]. The number of epochs was set to 10 as the ranked list of items hardly changes after 10 epochs. The regularization was proportional to the support of the given item/user/context. We did not use any other heuristics like time decay to focus on the pure performance of the algorithms. The weights in W were proportional to the number of events belonging to the given cell of the tensor.

We measured recall and precision on the $N = 1, \dots, 50$ interval. We consider items relevant to a user if the user has at least one event for that item in the test set. Recall@N is the ratio of relevant items on the ranked topN recommendations for the user relative to the number of the user's events in the test set. Precision@N is the ratio of the number of returned relevant items (for each user) and the number of total returned items. Greater values mean better performance.

Table 1 contains recall@20 values for every experiment. Recall@20 is important in practical application as the user usually sees maximum the top 20 items. Using context, the performance is increased overall. The selection of the appropriate context is crucial. In our experiments seasonality improved performance on all datasets. The sequentiality patterns caused large improvements on the Grocery and LastFM 1K datasets (significantly surpassed the results with the seasonality) but did not increased performance on the movie databases (VoD, Netflix, MovieLens). By including seasonality the performance is increased by an average of 30% for the VoD data. This agrees with our assumption that the VoD consumption has a very strong daily repetitiveness and the behavior in different time bands can be well segmented. The results increased by an additional 35% when we used iTALS instead of the context-aware baseline. The genre of the previously watched movies can also improve performance, however its extent is only around 10%. On the other two movie datasets iCA did not improve the performance significantly. We assume that this is due to the explicit–implicit transformation because the transformed implicit feedback is more reliable and also results a sparser tensor. The iTALS using seasonality however could achieve 30% and 80% improvement on Netflix and MovieLens respectively.

Inclusion of the sequentiality patterns increased the performance on Grocery and LastFM 1K datasets by more than 90% and 300% (compared to iALS, recall that no sequential iCA baseline is calculated). Interestingly, the model using the last category is the best with 20 features, but with 40 features the model using last two categories becomes better. We conjecture that this is connected to the greater expressive power of the model with more features. With seasonality the performance also improved by more than 50% and 75%, respectively, on these

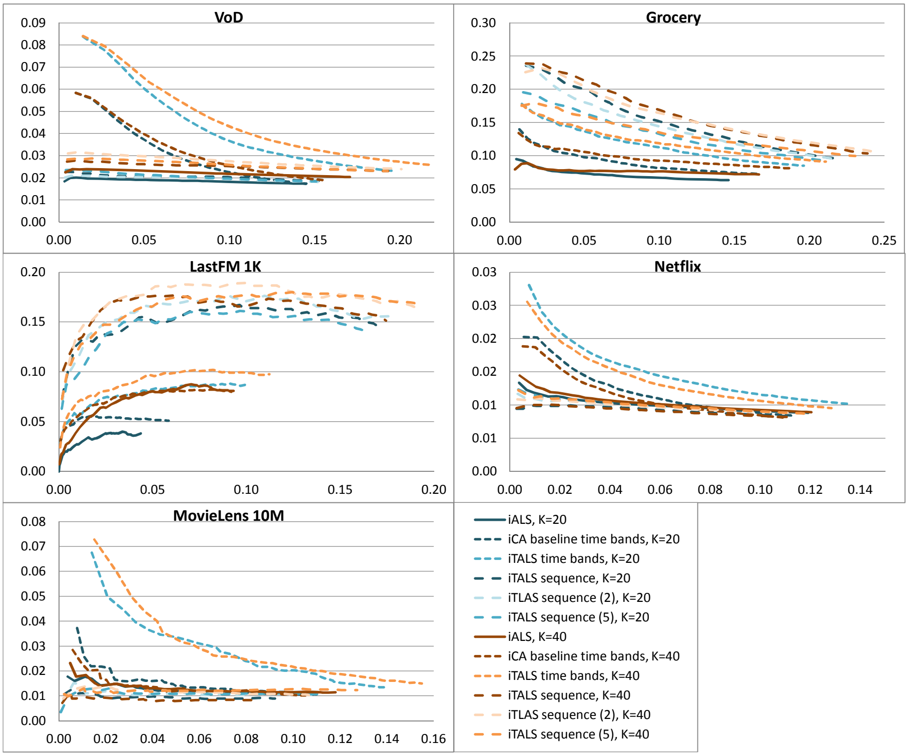


Fig. 1. Precision–recall curves for all datasets and algorithms using factorization with $K = 20$ (blue) and $K = 40$ (orange) factors. The y axis corresponds to precision and x to recall.

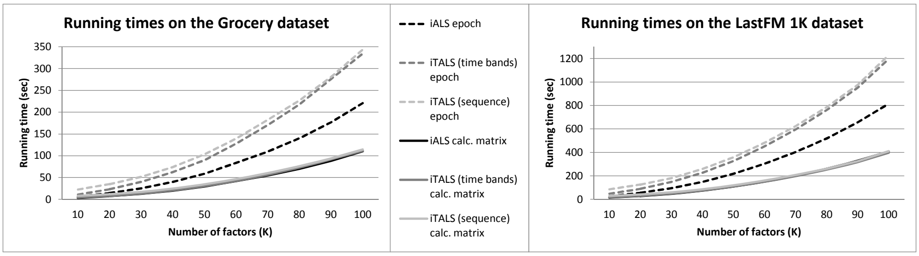


Fig. 2. Running times of iTALS compared to iALS on the Grocery and LastFM 1K datasets

datasets. We expected that the usage pattern learning will perform better on Grocery and LastFM 1K datasets than on the movie datasets as sequentiality is rather important in shopping and music listening than seasonality.

Figure 1 shows the precision–recall curves. The order of the performance of the algorithms is the same as with the recall@20. Observe that the distance between the curves of the iTALS variants and the curve of the iALS is larger when we use 40 features. Recall that the feature vectors of the context works as a reweighting of the user–item relation. If the resolution of this relation is finer, the reweighting can be more efficient and each factor describes a more specific item property, so the behavior in different context can be described more specifically. Thus, increasing the number of features results in larger performance increase for the context-aware iTALS variants than for iALS.

5.1 Running Times

We compared the running times of the iTALS and iALS algorithms in terms of K (see Figure 2). The experiments were run on a laptop with an Intel Core i5 2410M 2.3GHz processor using only one core. We depict only curves for 2 datasets, since others are similar. We made several runs for each K; the median of the epoch running times are shown (dashed lines). The solid lines show the computation time for one feature matrix. Observe that iTALS scales quadratically with K as iALS; the (re)computation time of one feature matrix is basically the same. Since iTALS recomputes more feature matrices its running time per epoch is larger. Importantly, even if the number of context-states is large (as with the sequential iTALS on LastFM 1K) the $O(K^2)$ part of the complexity remains dominant. This is because the number of non-zero elements in T is much larger than the number of different items/users/contex-states in every case where the usage of context-aware approaches is justified.

6 Conclusion

In this paper we presented an efficient ALS-based tensor factorization method for the context-aware implicit feedback recommendation problem. Our method, coined iTALS, scales linearly with the number of *non-zeroes* in the tensor, thus it works well on implicit data. We presented two specific examples for context-aware implicit scenario with iTALS. When using the seasonality as context, we efficiently segmented periodical user behavior in different time bands. When exploiting sequentiality in the data, the model was able to tell apart items having different repetitiveness in usage pattern. These variants of iTALS allow us to analyze user behavior by integrating arbitrary contextual information within the well-known factorization framework. Experiments performed on five large datasets show that proposed algorithms can greatly improve the performance. Compared to iALS and iCA, our algorithm attained an increase in recall@20 up to 300% and 35%.

One should, however, avoid creating a high dimensional tensors because the number of non-zero elements remains the same no matter how many context types are integrated; so tensors with more dimensions become sparser and thus the results may be poorer than with only a few context dimensions used. Our work opens up a new path for context-aware recommendations in the most common implicit feedback task when only the user history but no rating is available.

Future work will include the characterization of the relation between reweighting, context features and the number of features (K) as well as the design of further context-aware iTALS-based recommendation algorithms.

Acknowledgment. Authors thank Gábor Takács for his very useful comments on the paper.

References

1. Pilászy, I., Tikk, D.: Recommending new movies: Even a few ratings are more valuable than metadata. In: Recsys 2009: ACM Conf. on Recommender Systems, New York, NY, USA, pp. 93–100 (2009)
2. Bennett, J., Lanning, S.: The Netflix Prize. In: KDD Cup Workshop at SIGKDD 2007, San Jose, California, USA, pp. 3–6 (2007)
3. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: ICDM 2007: IEEE Int. Conf. on Data Mining, Omaha, NE, USA, pp. 43–52 (2007)
4. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Major components of the Gravity recommendation system. SIGKDD Explor. Newsl. 9, 80–83 (2007)
5. Pilászy, I., Zibriczky, D., Tikk, D.: Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In: Recsys 2010: ACM Conf. on Recommender Systems, Barcelona, Spain, pp. 71–78 (2010)
6. Takács, G., Pilászy, I., Tikk, D.: Applications of the conjugate gradient method for implicit feedback collaborative filtering. In: RecSys 2011: ACM Conf. on Recommender Systems, Chicago, IL, USA, pp. 297–300 (2011)
7. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: SIGKDD 2008: ACM Int. Conf. on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, pp. 426–434 (2008)
8. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems 20. MIT Press, Cambridge (2008)
9. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender Systems Handbook, pp. 1–35. Springer, US (2011)
10. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Recsys 2008: ACM Conf. on Recommender Systems, Lausanne, Switzerland, pp. 335–336 (2008)
11. Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver, N.: Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In: Recsys 2010: ACM Conf. on Recommender Systems, Barcelona, Spain, pp. 79–86 (2010)
12. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. 23(1), 103–145 (2005)
13. Adomavicius, G., Ricci, F.: Workshop on context-aware recommender systems (CARS-2009). In: Recsys 2009: ACM Conf. on Recommender Systems, New York, NY, USA, pp. 423–424 (2009)
14. Said, A., Berkovsky, S., De Luca, E.W.: Putting things in context: Challenge on context-aware movie recommendation. In: CAMRa 2010: Workshop on Context-Aware Movie Recommendation, Barcelona, Spain, pp. 2–6 (2010)

15. Bogers, T.: Movie recommendation using random walks over the contextual graph. In: CARS 2010: 2nd Workshop on Context-Aware Recommender Systems, Barcelona, Spain, pp. 1–5 (2010)
16. Baltrunas, L., Amatriain, X.: Towards time-dependant recommendation based on implicit feedback. In: CARS 2009: Workshop on Context-aware Recommender Systems, New York, NY, USA, pp. 1–5 (2009)
17. Bader, R., Neufeld, E., Woerndl, W., Prinz, V.: Context-aware POI recommendations in an automotive scenario using multi-criteria decision making methods. In: CaRR 2011: Workshop on Context-awareness in Retrieval and Recommendation, Palo Alto, CA, USA, pp. 23–30 (2011)
18. He, Q., Pei, J., Kifer, D., Mitra, P., Giles, L.: Context-aware citation recommendation. In: WWW 2010: Int. Conf. on World Wide Web, Raleigh, NC, USA, pp. 421–430 (2010)
19. Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., Pedone, A.: Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In: Recsys 2009: ACM Conf. on Recommender Systems, New York, NY, USA, pp. 265–268 (2009)
20. Lathauwer, L.D., Moor, B.D., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. 21(4), 1253–1278 (2000)
21. Rendle, S., Gantner, Z., Freudenthaler, C., Schmidt-Thieme, L.: Fast context-aware recommendations with factorization machines. In: SIGIR 2011: ACM Int. Conf. on Research and Development in Information, Beijing, China, pp. 635–644 (2011)
22. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: ICDM 2008: IEEE Int. Conf. on Data Mining, Pisa, Italy, pp. 263–272 (2008)
23. Liu, N.N., Cao, B., Zhao, M., Yang, Q.: Adapting neighborhood and matrix factorization models for context aware recommendation. In: CAMRa 2010: Workshop on Context-Aware Movie Recommendation, Barcelona, Spain, pp. 7–13 (2010)
24. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD 1993: ACM SIGMOD Int. Conf. on Management of Data, Washington DC, USA, pp. 207–216 (1993)
25. Davidson, J., Liebald, B., Liu, J., et al.: The YouTube video recommendation system. In: Recsys 2010: ACM Conf. on Recommender Systems, Barcelona, Spain, pp. 293–296 (2010)
26. Celma, O.: Music Recommendation and Discovery in the Long Tail. Springer (2010)
27. GroupLens Research: Movielens data sets (2006), http://www.grouplens.org/node/73

Probability Estimation for Multi-class Classification Based on Label Ranking

Weiwei Cheng and Eyke Hüllermeier

Mathematics and Computer Science Department
University of Marburg, Germany
{cheng,eyke}@mathematik.uni-marburg.de

Abstract. We consider the problem of probability estimation in the setting of multi-class classification. While this problem has already been addressed in the literature, we tackle it from a novel perspective. Exploiting the close connection between probability estimation and ranking, our idea is to solve the former on the basis of the latter, taking advantage of recently developed methods for label ranking. More specifically, we argue that the Plackett-Luce ranking model is a very natural choice in this context, especially as it can be seen as a multinomial extension of the Bradley-Terry model. The latter provides the basis of pairwise coupling techniques, which arguably constitute the state-of-the-art in multi-class probability estimation. We explore the relationship between the pairwise and the ranking-based approach to probability estimation, both formally and empirically. Using synthetic and real-world data, we show that our method does not only enjoy nice theoretical properties, but is also competitive in terms of accuracy and efficiency.

1 Introduction

The problem of classification is normally understood as learning a model that maps instances to class labels. While useful for many purposes, there are numerous applications in which the estimation of the probabilities of the different classes is more desirable than just selecting one of them. Application areas of this kind include safety-critical domains such as medical decision making, where probabilities are useful as a measure of the reliability of a classification, or applications like computational advertising, where they allow one to focus on the most promising alternatives. Moreover, given a probability for each class, it is in principle possible to minimize any loss function, that is, to derive (or at least approximate) Bayes-optimal decisions. This is especially useful in cost-sensitive classification, where different types of misclassification may incur different costs [12]. Simply minimizing the standard 0/1 loss will normally not produce desirable results in this setting.

As discussed in more detail in Section 2, the problem of probability estimation is rather challenging and in a sense even more difficult than conventional classification. In the field of machine learning, the problem has been approached from different directions. Specifically interesting in this regard is the idea of exploiting the connection between probability estimation and *ranking*, another type of

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 83–98, 2012.

prediction problem that has attracted increasing attention in recent years. Indeed, ranking is in a sense in-between classification and probability estimation or, stated differently, can be seen as an intermediate step from classification to probability estimation [8]. In particular, the maximization of ranking measures like the AUC requires sorting a given set of alternatives from most probable positive to most probable negative [5]. Thus, although precise probability degrees of all alternatives are not necessarily needed, at least their order relation must be predicted correctly.

For far, most work on the connection between probability estimation and ranking, including AUC maximization, has focused on the *binary* case, distinguishing only between two classes (positive and negative). Essentially, this means that only a single value needs to be estimated for each instance, namely the probability of belonging to the positive class. In this paper, we establish a connection between probability estimation and ranking for the case of *multiple classes.* To this end, we refer to another type of ranking problem, namely *label ranking* [3,4]. While the binary case is intimately connected with *bipartite ranking*, in which the *instances* are ranked themselves, the problem of label ranking consists of ranking the *class labels* given an instance.

The rest of the paper is organized as follows. In the next section, we discuss the problem of multi-class probability estimation and recall the basic ideas of pairwise coupling and classifier calibration. In Section 3, we introduce the problem of label ranking. Then, in Section 4, we establish a tight link between label ranking and probability estimation, taking advantage of a probabilistic ranking model called Plackett-Luce. In Section 5, we show how the label ranking problem can be approached on the basis of this model. Building on the connection established in Section 4 and the PL-based label ranking method introduced in Section 5, we then introduce a method for probability estimation based on label ranking in Section 6. Experimental results are presented in Section 7, before concluding the paper in Section 8.

2 Multi-class Probability Estimation

Consider the standard setting of multi-class classification with an instance space $\mathbb{X}$ and a set of classes $\mathcal{Y} = \{y_1, \ldots, y_K\}$. We are interested in learning a probabilistic classifier, that is, a model that estimates the conditional probabilities of classes given an instance $\boldsymbol{x} \in \mathbb{X}$:

$$(p_1, \ldots, p_K) = (\mathbf{P}_{\mathcal{Y}}(y_1 \,|\, \boldsymbol{x}), \ldots, \mathbf{P}_{\mathcal{Y}}(y_K \,|\, \boldsymbol{x})) \tag{1}$$

Since true probability degrees are rarely available for training, probabilistic classifiers are typically trained on standard classification data, that is, observations of the form $(\boldsymbol{x}, y) \in \mathbb{X} \times \mathcal{Y}$, where the class label y is assumed to be generated according to $\mathbf{P}_{\mathcal{Y}}(\cdot \,|\, \boldsymbol{x})$.

Probability estimation is known to be a quite hard problem, especially in comparison to standard classification. This comes at no surprise, noting that proper probability estimation is a sufficient but not necessary condition for proper classification: If the conditional class probabilities (1) are predicted accurately, an optimal classification can simply be made by picking the class with highest probability:

$$\hat{y} = \arg\max_{y_i \in \mathcal{Y}} \hat{\mathbf{P}}(y_i \,|\, \boldsymbol{x}) \tag{2}$$

More generally, the Bayes decision can be taken so as to minimize any loss in expectation. On the other hand, a correct classification can also be obtained based on less accurate probability estimates. In fact, the classification will remain correct as long as the estimated probability is highest for the true class. Or, stated differently, an estimation error will remain ineffective unless it changes the result of the arg max operation in (2). This is also the reason for why methods like naive Bayes show competitive performance in classification despite producing relatively inaccurate probability estimates [7].

Methods like naive Bayes and decision trees are multi-class classifiers and can in principle be used to produce probability estimates in this setting. In practice, however, one often prefers to estimate probabilities in the two-class setting, especially because estimating a single probability (of the positive class) is much simpler than estimating $K-1$ probabilities simultaneously. Moreover, the binary case is amenable to a broader spectrum of classifiers, including logistic regression, which is a proven method for probability estimation. On the other hand, the reduction of multinomial to binomial probability estimation obviously involves an aggregation problem, namely the need to combine probabilities on pairs of classes into probabilities on the label set $\mathcal{Y}$. This is the idea of "pairwise coupling" techniques.

2.1 Pairwise Coupling

As a special type of binary decomposition technique, pairwise coupling allows one to tackle multi-class problems with binary classifiers. The key idea is to transform a K-class problem into $K(K-1)/2$ binary problems, one for each pair of classes. More specifically, a separate model $M_{i,j}$ is trained for each pair of labels $(y_i, y_j) \in \mathcal{Y} \times \mathcal{Y}$, $1 \leq i < j \leq K$, using the examples from these two classes as their training set; thus, a total number of $K(K-1)/2$ models is needed. $M_{i,j}$ is intended to separate the objects with label y_i from those having label y_j.

At prediction time, a query instance $\boldsymbol{x} \in \mathbb{X}$ is submitted to all models $M_{i,j}$. The predictions $p_{i,j} = M_{i,j}(\boldsymbol{x})$ are typically interpreted by means of the Bradley-Terry model [1], a probabilistic choice model expressing the probability that "y_i wins against y_j" as follows:

$$p_{i,j} = \mathbf{P}(y_i \succ y_j) = \mathbf{P}_{\mathcal{Y}}(y_i \,|\, \{y_i, y_j\}) = \frac{p_i}{p_i + p_j} \tag{3}$$

Based on the relationship (3), the unconditional probabilities p_i can be derived from the conditional (pairwise) probabilities $p_{i,j}$. Obviously, however, it will not always be possible to find a distribution $(p_1, \ldots, p_K)$ such that the equality

$p_{i,j} = p_i/(p_i + p_j)$ holds for all $1 \leq i < j \leq K$, simply because this system of equations is over-constrained: K variables have to satisfy $K(K-1)/2$ equations (plus the constraint $p_1 + \ldots + p_K = 1$). In fact, one should notice that the models $M_{i,j}$ are learnt independently of each other, so that the predictions $p_{i,j}$ are not necessarily coherent.

Pairwise coupling techniques therefore seek to solve the above reconstruction problem approximately. Different methods for putting this idea into practice have been proposed and compared in [21]. For example, the following system of linear equations can be derived by "averaging" over the identities $\mathbf{P}_{\mathcal{Y}}(y_i) = \mathbf{P}_{\mathcal{Y}}(y_i \,|\, \{y_i, y_j\}) \cdot \mathbf{P}_{\mathcal{Y}}(\{y_i, y_j\})$:

$$\begin{aligned}\mathbf{P}_{\mathcal{Y}}(y_i) &= \frac{1}{K-1} \sum_{j \neq i} \mathbf{P}_{\mathcal{Y}}(y_i \,|\, \{y_i, y_j\}) \cdot \mathbf{P}_{\mathcal{Y}}(\{y_i, y_j\}) \\ &= \frac{1}{K-1} \sum_{j \neq i} \mathbf{P}_{\mathcal{Y}}(y_i \,|\, \{y_i, y_j\}) \cdot (\mathbf{P}_{\mathcal{Y}}(y_i) + \mathbf{P}_{\mathcal{Y}}(y_j))\end{aligned}$$

Or, in terms of the p_i an $p_{i,j}$:

$$(K-1)p_i \;=\; \sum_{j \neq i} p_{i,j} \cdot (p_i + p_j)$$

In conjunction with the constraint $p_1 + \ldots + p_K = 1$ and the non-negativity of the p_i, this system has a unique solution provided that $p_{i,j} > 0$ for all $1 \leq i, j \leq K$. Among the methods compared in [21], this approach turned out to perform specifically well.

2.2 Classifier Calibration

As mentioned earlier, the scores produced by conventional classification methods are typically quite biased: Although they might be good enough for correct classification, they do not provide accurate probability estimates. This is true even for methods with an inherently probabilistic interpretation, such as logistic regression [24]. Among a number of possible reasons, let us mention that all such methods are based on rather strong model assumptions that will commonly be violated in practice. In naive Bayes, for example, this is the assumption of conditional independence of the attributes given the class. Likewise, logistic regression assumes that the log of the odds ratio is a linear function of the attributes. Another reason is the fact that for commonly used loss functions such as 0/1 loss or hinge loss, the true probability is *not* a risk minimizer [2].

To overcome this problem, several methods for "classifier calibration" have been proposed in the literature [18,22]. These are post-processing methods whose idea is to find a mapping that turns classifier scores into meaningful probability estimates. As an example, we mention the method of isotonic regression [23], which, due to its nature as a nonparametric approach, is less susceptible to the aforesaid problem of model misspecification. Besides, it has been shown to perform quite well in practice [16].

Isotonic regression finds a monotone mapping from scores to probabilities or, say, from poorly estimated probabilities to hopefully better ones. Monotonicity assures that the order of classes can never be reversed: If class y_i receives a higher score by the classifier than y_j, then the calibrated probability estimate for the former cannot be smaller than the estimate for the latter. Against the background of our discussion about the relationship between ranking and probability estimation, this is clearly a desirable property.

3 Label Ranking

In the setting of label ranking, each instance $\boldsymbol{x}$ from the instance space $\mathbb{X}$ is associated with a total order of all class labels, that is, a total, transitive, and asymmetric relation $\succ_{\boldsymbol{x}}$ on $\mathcal{Y}$, where $y_i \succ_{\boldsymbol{x}} y_j$ indicates that y_i precedes y_j in the order. Since a ranking can be considered as a special type of preference relation, we shall also say that $y_i \succ_{\boldsymbol{x}} y_j$ indicates that y_i is *preferred* to y_j given the instance $\boldsymbol{x}$.

Formally, a total order $\succ_{\boldsymbol{x}}$ can be identified with a permutation $\pi_{\boldsymbol{x}}$ of the set $[K] = \{1, \ldots, K\}$. We define $\pi_{\boldsymbol{x}}$ such that $\pi_{\boldsymbol{x}}(i)$ is the index j of the class label y_j put on the i-th position in the order (and hence $\pi_{\boldsymbol{x}}^{-1}(j) = i$ the position of the j-th label). This permutation thus encodes the (ground truth) order relation

$$y_{\pi_{\boldsymbol{x}}(1)} \succ_{\boldsymbol{x}} y_{\pi_{\boldsymbol{x}}(2)} \succ_{\boldsymbol{x}} \cdots \succ_{\boldsymbol{x}} y_{\pi_{\boldsymbol{x}}(K)} \enspace .$$

The class of permutations of $[K]$ (the symmetric group of order K) is denoted by Ω. By abuse of terminology, though justified in light of the above one-to-one correspondence, we refer to elements $\pi \in \Omega$ as both permutations and rankings.

In analogy with the classification setting, we do not assume the existence of a deterministic $\mathbb{X} \longrightarrow \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω. This means that, for each $\boldsymbol{x} \in \mathbb{X}$, there exists a probability distribution $\mathbf{P}_\Omega(\cdot \,|\, \boldsymbol{x})$ such that, for every $\pi \in \Omega$, $\mathbf{P}_\Omega(\pi \,|\, \boldsymbol{x})$ is the probability that $\pi_{\boldsymbol{x}} = \pi$.

The goal in label ranking is to learn a "label ranker" in the form of an $\mathbb{X} \longrightarrow \Omega$ mapping. As training data, a label ranker uses a set of instances $\boldsymbol{x}_n$, $n \in [N]$, together with information about the associated rankings $\pi_{\boldsymbol{x}_n}$. Ideally, complete rankings are given as training information. From a practical point of view, however, it is important to allow for incomplete information in the form of a ranking

$$y_{\pi_{\boldsymbol{x}}(1)} \succ_{\boldsymbol{x}} y_{\pi_{\boldsymbol{x}}(2)} \succ_{\boldsymbol{x}} \cdots \succ_{\boldsymbol{x}} y_{\pi_{\boldsymbol{x}}(k)} \enspace , \tag{4}$$

where $k < K$ and $\{\pi(1), \ldots, \pi(k)\} \subset [K]$. For example, for an instance $\boldsymbol{x}$, it might be known that $y_2 \succ_{\boldsymbol{x}} y_1 \succ_{\boldsymbol{x}} y_5$, while no preference information is given about the labels y_3 or y_4. By definition, we let $\pi^{-1}(y_i) = \pi^{-1}(i) = 0$ if y_i is not present in the ranking π; thus, the presence of a class y_i is equivalent to $\pi^{-1}(i) > 0$.

Table 1. A distribution of rankings with three labels

π	$\mathbf{P}_{\Omega}(\pi \mid \boldsymbol{x})$
$y_1 \succ y_2 \succ y_3$	0.10
$y_1 \succ y_3 \succ y_2$	0.25
$y_2 \succ y_1 \succ y_3$	0.20
$y_2 \succ y_3 \succ y_1$	0.20
$y_3 \succ y_1 \succ y_2$	0.25
$y_3 \succ y_2 \succ y_1$	0

4 Label Ranking vs Classification: A Probabilistic Link

In contrast to conventional classification, the setting of label ranking does not assume the existence of a "true class label" of an instance. In fact, while the output space in classification is given by the set $\mathcal{Y}$ of class labels, and a probability vector of conditional class probabilities (1) can be associated with every instance $\boldsymbol{x} \in \mathcal{X}$, the output space in label ranking is the class of permutations Ω. Yet, as will be explained in the following, label ranking can be interpreted as a generalization of conventional classification or, the other way around, classification can be seen as a special case of label ranking. Most naturally, this connection is obtained by associating the "true class" in classification with the top-ranked label in label ranking. For the ease of exposition, we shall subsequently drop the conditioning on the instance $\boldsymbol{x}$.

4.1 From Probabilities on Rankings to Class Probabilities

Formally, the connection between label ranking and classification is established by means of a mapping between the spaces $\mathfrak{P}(\mathcal{Y})$ and $\mathfrak{P}(\Omega)$, that is, the space of probability distributions on $\mathcal{Y}$ and the space of probability distributions on Ω. Associating the observed class in classification with the top-ranked label in label ranking then comes down to mapping a measure $\mathbf{P}_{\Omega} \in \mathfrak{P}(\Omega)$ to a measure $\mathbf{P}_{\mathcal{Y}} \in \mathfrak{P}(\mathcal{Y})$ such that

$$p_j = \mathbf{P}_{\mathcal{Y}}(y_j) = \sum_{\pi \in \Omega :\, \pi(1) = j} \mathbf{P}_{\Omega}(\pi) \ . \tag{5}$$

For example, the probability distribution $\mathbf{P}_{\Omega}$ in Table 1 is mapped to the distribution $\mathbf{P}_{\mathcal{Y}} = (p_1, p_2, p_3) = (0.35, 0.4, 0.25)$. Note that the most probable class (y_2) differs from the top-label in the most probable ranking (y_1).

The other way around, there are several ways of embedding $\mathfrak{P}(\mathcal{Y})$ in $\mathfrak{P}(\Omega)$ (indeed, note that $|\Omega|$ is in general much larger than $|\mathcal{Y}|$); we will come back to this issue when discussing the so-called Plackett-Luce model below.

4.2 The Plackett-Luce Model

So far, no specific assumptions about the probability measure $\mathbf{P}_{\Omega}$ on Ω were made. Needless to say, due to the large cardinality of the space Ω, it is practically impossible to work with the full class of distributions $\mathfrak{P}(\Omega)$. For that reason, different types of *parametrized* classes of probability distributions on rankings have been proposed in statistics [15].

A prominent example is the Mallows model [14], a *distance-based* probability model belonging to the family of exponential distributions. The standard Mallows model is determined by two parameters:

$$\mathbf{P}_{\Omega}(\pi \,|\, \theta, \pi_0) = \frac{\exp(-\theta D(\pi, \pi_0))}{\phi(\theta)} \tag{6}$$

The ranking $\pi_0 \in \Omega$ is the location parameter (mode, center ranking) and $\theta \geq 0$ is a spread parameter. Moreover, $D(\cdot)$ is a distance measure on rankings, and the constant $\phi = \phi(\theta)$ is a normalization factor that depends on the spread (but, provided the right-invariance of $D(\cdot)$, not on π_0).

In the following, we shall focus on another model that was first studied by Luce [13] and subsequently by Plackett [17]. The Plackett-Luce (PL) model appears to be especially appealing for our purpose, as it establishes a natural bridge between label ranking and classification. The PL model is specified by a parameter vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_K) \in \mathbb{R}_+^K$:

$$\mathbf{P}_{\Omega}(\pi \,|\, \boldsymbol{v}) \; = \prod_{i=1}^{K} \frac{v_{\pi(i)}}{v_{\pi(i)} + v_{\pi(i+1)} + \ldots + v_{\pi(K)}} \tag{7}$$

Obviously, this model can be seen as a generalization of the above-mentioned Bradley-Terry model (3) for the pairwise comparison of alternatives. Indeed, a natural interpretation of the PL model is a *stage-wise* construction of a ranking: A ranking is produced by a sequence of *choices*, where each choice problem consists of selecting one among the labels that have not been picked so far, and the probability of a label y_i being selected is always proportional to its "skill" parameter v_i. First, the top label is chosen, and the probability of each label y_i to be selected is given by $v_i/(v_1 + v_2 + \ldots + v_K)$. Then, the second label is chosen among those still available, using the same selection principle, and so on and so forth. In other words, with probabilities p_i defined as "normalized skills"

$$p_i = \mathbf{P}_{\mathcal{Y}}(y_i) = \frac{v_i}{v_1 + v_2 + \ldots + v_K} \;\;, \tag{8}$$

the probability of y_i to be chosen among a set $C \subseteq \mathcal{Y}$ of remaining candidates exactly equals the conditional probability $\mathbf{P}_{\mathcal{Y}}(y_i \,|\, C)$. Consequently, the probability (7) can also be written as follows:

$$\mathbf{P}_{\Omega}(\pi \,|\, \boldsymbol{v}) = \mathbf{P}_{\Omega}(\pi \,|\, \boldsymbol{p}) \; = \prod_{i=1}^{K} \mathbf{P}_{\mathcal{Y}} \left(y_{\pi(i)} \,|\, C_i \right) \;\;, \tag{9}$$

where $C_i = \{y_{\pi(i)}, \ldots, y_{\pi(K)}\}$ is the set of remaining candidates and $\boldsymbol{p} = \boldsymbol{v}/||\boldsymbol{v}||$ is the probability vector obtained by normalizing the parameter vector $\boldsymbol{v}$.

Thus, with a PL model $\boldsymbol{v} = (v_1, \ldots, v_K)$, one can simultaneously associate a distribution $\mathbf{P}_{\mathcal{Y}}$ on $\mathcal{Y}$ and a distribution $\mathbf{P}_{\Omega}$ on Ω that are closely connected with each other. In particular, this model is coherent with the mapping (5) in the sense that $\mathbf{P}_{\mathcal{Y}}(y_i) = \mathbf{P}_{\Omega}(\pi(1) = i)$. Moreover, the PL model defines a specific though natural embedding of $\mathfrak{P}(\mathcal{Y})$ in $\mathfrak{P}(\Omega)$ via (9). Last but not least, it allows for computing the probability of incomplete rankings (which normally requires an expensive marginalization, i.e., summation over all linear extensions) in a quite convenient way: The probability of an incomplete ranking (4) is given by

$$\mathbf{P}(\pi \,|\, \boldsymbol{v}) \;=\; \prod_{i=1}^{k} \frac{v_{\pi(i)}}{v_{\pi(i)} + v_{\pi(i+1)} + \ldots + v_{\pi(k)}} \enspace .$$

As an aside, we mention that the appealing properties of the PL model as outlined above are closely connected with the "choice axioms" of Luce [13]. In fact, it is known that the PL model is the only ranking model satisfying these axioms.

5 Label Ranking Based on the PL Model

A label ranking method based on the PL model has been proposed in [3]. The key idea of this method is to define the PL parameters $\boldsymbol{v}$ as a function of the input attributes specifying an instance: $\boldsymbol{v} = (v_1, \ldots, v_K) = f(\boldsymbol{x})$. More specifically, log-linear models are used to guarantee non-negativity, that is, the logarithm of each parameter v_i is modeled as a linear function:

$$v_i \;=\; \exp\left(\langle \boldsymbol{w}^{(i)}, \boldsymbol{x} \rangle\right) \;=\; \exp\left(\sum_{j=1}^{d} w_j^{(i)} \cdot x_j\right) \enspace , \tag{10}$$

where an instance is assumed to be represented in terms of a feature vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathcal{X} \subseteq \mathbb{R}^d$.

5.1 Parameter Estimation

Learning a label ranking model then comes down to estimating the parameters $w_j^{(i)}$ $(i \in [K], j \in [d])$ in (10). This can be accomplished by means of maximum likelihood estimation. More precisely, given a training data set

$$\mathcal{T} \;=\; \left\{ \left(\boldsymbol{x}^{(q)}, \pi^{(q)} \right) \right\}_{q=1}^{N} \tag{11}$$

with $\boldsymbol{x}^{(q)} = \left(x_1^{(q)}, \ldots, x_d^{(q)} \right)$, the parameters are determined by maximizing the log-likelihood function

$$L = \sum_{q=1}^{N} \sum_{i=1}^{n_q} \left[\log v\left(\pi^{(q)}(i), q\right) - \log \sum_{j=i}^{n_q} v\left(\pi^{(q)}(j), q\right) \right] \, , \tag{12}$$

where n_q is the number of labels in the ranking $\pi^{(q)}$, and

$$v(i,q) \;=\; \exp\left(\langle \boldsymbol{w}^{(i)}, \boldsymbol{x}^{(q)} \rangle \right) \enspace . \tag{13}$$

For algorithmic details, we refer to [3].

5.2 From Label Ranking to Logistic Regression

Interestingly, we can show that the standard multinomial logistic regression approach for classification can be seen as a special case of the PL-based label ranking method introduced above. To this end, consider the case of classification, where a single class label is observed for each training instance. An observation of this kind can be interpreted as a label ranking, of which only the top-position is known. Or, stated differently, the probability of this observation corresponds to the probability of selecting y_i in the first step of the choice process:

$$\mathbf{P}_{\mathcal{Y}}(y_i \,|\, \boldsymbol{x}; \boldsymbol{w}) = \mathbf{P}_{\Omega}(\pi(1) = i \,|\, \boldsymbol{x}; \boldsymbol{w}) = \frac{\exp\left(\langle \boldsymbol{w}^{(i)}, \boldsymbol{x} \rangle\right)}{\sum_{j=1}^{K} \exp\left(\langle \boldsymbol{w}^{(j)}, \boldsymbol{x} \rangle\right)} \tag{14}$$

The log-likelihood function of the data is then given by

$$L = \sum_{q=1}^{N} \sum_{i=1}^{K} t_{qi} \left[\langle \boldsymbol{w}^{(i)}, \boldsymbol{x} \rangle - \log \sum_{j=1}^{K} \exp\left(\langle \boldsymbol{w}^{(j)}, \boldsymbol{x} \rangle \right) \right] \ , \tag{15}$$

where $\boldsymbol{t}$ is a coding matrix with $t_{qi} = 1$ if the class of the q-th instance is y_i and $t_{qi} = 0$ otherwise. This model exactly corresponds to the standard model of multinomial logistic regression.

6 A Ranking Approach to Probability Estimation

In our discussion so far, we have established a close connection between (label) ranking and classification. In terms of *modeling*, this connection mainly rests on the interpretation of a classification (an observed class label) as a ranking with the top-label observed. This connection is ideally supported by the PL model, notably because the ranking parameters of this model are in direct correspondence with class probabilities; besides, probabilities of incomplete rankings (i.e., rankings of a subset of the labels) are obtained through simple conditioning. As a consequence, the PL model is also consistent with our monotonicity assumption: The higher the class probability, the higher the (expected) position of the corresponding label in the ranking.

In terms of *methods*, we have noticed that label ranking based on the PL model can be seen as an extension of conventional multinomial logistic regression; or, vice versa, logistic regression corresponds to a special case of PL-based label ranking, namely the case where only top-1 rankings (classes) are observed. The obvious advantage of the label ranking framework is an increased flexibility with regard to the exploitation of training information: While standard logistic regression can only learn from observed class labels, label ranking is also able to exploit comparative preference information of more general type. This includes, for example, pairwise comparisons of the kind "for the instance $\boldsymbol{x}$, class label y_i is more probable than y_j", even if one cannot assure that y_i is the most likely label. This could be useful in many practical situations, for example if the correct class label cannot be determined precisely although some candidate classes can certainly be excluded [6].

More generally, a ranking can be interpreted as a special type of *qualitative probability* on $\mathcal{Y}$ [20]. The order relation $y_i \succ_{\boldsymbol{x}} y_j$ indicates that the conditional probability of y_i given $\boldsymbol{x}$ is higher than the probability of y_j given $\boldsymbol{x}$, though without specifying any concrete numerical values. By learning a PL-based label ranking model, these qualitative probabilities are then turned into quantitative probabilities $p_i \propto v_i(\boldsymbol{x})$. Thus, label ranking can indeed be seen as a natural bridge between classification and probability estimation.

6.1 PELARA: Probability Estimation via Label Ranking

Our method of Probability Estimation via LAbel RAnking (PELARA) can be summarized as follows:

- The method assumes as training information a set of data (11) consisting of instances $\boldsymbol{x} \in \mathbb{X}$ together with label rankings (4) of varying length $k \in [K]$ (including $k = 1$ for the special case of a class observation).
- On this data, a label ranker is trained using the method outlined in Section 5 (and explained in more detail in [3]).
- As a result, we obtain a model M' that assigns a vector of PL parameters to each query instance $\boldsymbol{x}$:

$$M' : \boldsymbol{x} \mapsto \boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}) \in \mathbb{R}_+^K$$

- To obtain an (uncalibrated) probability estimate, these vectors are normalized, i.e., $\boldsymbol{v}(\boldsymbol{x})$ is turned into

$$M(\boldsymbol{x}) = \boldsymbol{p}(\boldsymbol{x}) = (p_1(\boldsymbol{x}), \ldots, p_K(\boldsymbol{x})) \propto \boldsymbol{v}(\boldsymbol{x}) \tag{16}$$

  such that $||\boldsymbol{p}(\boldsymbol{x})|| = 1$.

The model M thus obtained defines a probability estimator.

6.2 Comparison with Decomposition Schemes

PELARA offers an appealing alternative to conventional methods such as pairwise coupling. Instead of decomposing the problem into a quadratic number of binary problems first, and combining the predictions of the pairwise models afterward, our label ranking method solves the original problem in one go. As a potential advantage, apart from simplicity, let us mention that the scores (probabilities) thus produced should be well-balanced right away, without the need to couple them in an approximate manner.

Indeed, one should notice that a pairwise decomposition will normally come with a loss of information, and the underlying assumptions justifying the reduction are not entirely clear. For example, while in our approach, the observation of class y_i for an instance $\boldsymbol{x}$ is modeled in terms of the probability $v_i/(v_1+\ldots+v_K)$, it is split into $K-1$ binary training examples $y_i \succ y_j, j \in [K]\backslash\{i\}$, in the pairwise approach. However, selecting y_i among the set of candidates $\mathcal{Y}$ is obviously not the same as (independently) selecting y_i in the pairwise comparisons between y_i and y_j:

$$\frac{v_i}{v_1+\ldots+v_K} \neq \prod_{j\neq i} \frac{v_i}{v_i+v_j}$$

In the case of a uniform distribution $\boldsymbol{v} \equiv 1$, for instance, the left-hand side is $1/K$ while the right-hand side is $(1/2)^{K-1}$. Similar arguments apply to the decomposition of an observed ranking into pairwise preferences.

The most common alternative to the pairwise (all pairs) decomposition scheme is one-vs-rest (OVR) decomposition [19]: One model is trained for each class label y_i, using this label as positive and all others as negative examples; for probability estimation, the predictions of these models are simply normalized. Thus, OVR trains a smaller number of models. The individual models, however, are typically more complex: Separating a class from all other class simultaneously is normally more difficult than only separating it from each class individually, and consequently may call for more complex decision boundaries. Besides, the individual problems may become quite imbalanced.

Our approach is in a sense in-between pairwise and OVR learning: Like OVR, it trains a linear number of models, one for each label. Yet, since these models are all trained simultaneously, without building negative meta-classes, the aforesaid disadvantage of OVR is avoided.

7 Experiments

This section presents experimental results, starting with a simplified analysis that is meant to help understand some key differences between the pairwise coupling and the ranking-based approach to probability estimation. Next, we compare our method with state-of-the-art approaches to probability estimation on a set of classification benchmarks.

7.1 On the Reconstruction Error of Pairwise Coupling

In comparison to the pairwise approach to probability estimation, which consists of decomposing the original multi-class problem into binary problems first and "coupling" the solutions (probability estimates) of these problems afterward, our ranking-based method allows for solving the original problem in a single step: Since all labels are treated simultaneously, there is no need for any type of aggregation. In principle, this should be seen as an advantage, especially since the decomposition step in pairwise learning is supposed to come along with a loss of information.

More concretely, one may wonder to what extent pairwise coupling is able to reconstruct a probability vector $\boldsymbol{p} = (p_1, \ldots, p_K)$ from its pairwise components

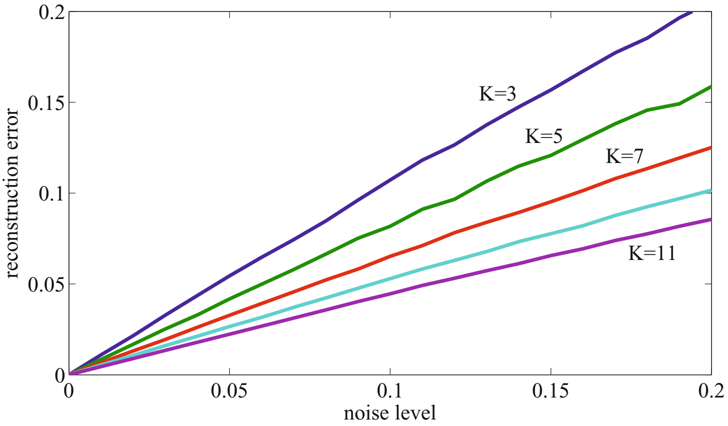


Fig. 1. Reconstruction error (measured in terms of RMSE) of pairwise coupling as a function of the level of noise (standard deviation) in the pairwise predictions; K corresponds to the number of class labels

$p_{i,j} = p_i/(p_i + p_j)$ if these are corrupted with noise; this ability is in fact crucial, since these pairwise components correspond to the predictions of the binary classifiers, which are never perfect.

Fig. 1 shows the expected RMSE between the true probability vector $\boldsymbol{p}$ and its coupled reconstruction (based on the method described in Section 2.1) when the pairwise probability estimates are given by the $p_{i,j}$ independently corrupted with additive Gaussian noise (truncated if necessary, so as to assure values in $[0,1]$). More specifically, the figure shows the expected RMSE as a function of the noise level, measured in terms of the standard deviation. As can be seen, the reconstruction error does indeed increase almost linearly with the noise level. What is also interesting to observe, however, is that the error becomes smaller if the number of classes increases. This effect, which has also been observed for other types of pairwise learning methods, can be explained by the level of redundancy produced by the pairwise approach: Since the number of models (and hence the number of prediction errors) increases quadratically, there is a good chance to "average out" the individual prediction errors.

On the other side, the ranking-based approach will of course be affected by prediction errors, too. These errors are not easily comparable to the pairwise ones, but suppose that we add the same Gaussian noise to the individual components p_i of the vector $\boldsymbol{p}$. The "reconstruction" in this case simply comes down to renormalization. Fig. 2 plots the corresponding reconstruction error r against the reconstruction error r' of the pairwise coupling approach; the circles in this picture are centered at the points (r, r'), where both r and r' refer to the same underlying (true) probability vector. Interestingly, while the ranking-based approach seems to have an advantage in the case of a low number of labels (the cloud of circles for four labels is above the diagonal), this advantage turns into a disadvantage if the number of labels increases. Indeed, the larger the number of

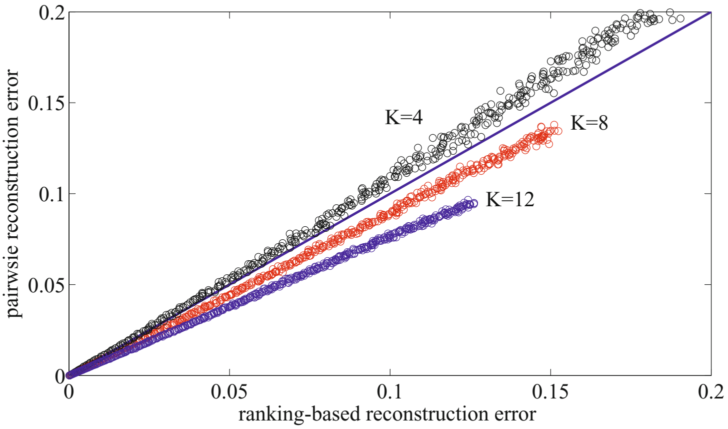


Fig. 2. Reconstruction error of the ranking-based approach (x-axis) versus reconstruction error of pairwise coupling (y-axis) for $K = 4, 8$ and 12 labels

labels, the more advantageous the pairwise coupling approach becomes; in the figure, the cases of eight and twelve labels are shown for illustration.

Needless to say, these results have to be interpreted with caution, since they are based on very idealized assumptions (e.g., independence of errors). Yet, they confirm an observation that was already made in previous studies of pairwise learning (albeit related to classification, not probability estimation): Due to the large number of binary models constructed, coming along with a high level of redundancy, the pairwise decomposition technique exhibits a kind of error-correction mechanism, and the larger the number of classes, the better this mechanism works [10].

7.2 Multi-class Classification

In the absence of benchmark data with given probabilities as ground-truth, we test our approach on standard classification benchmarks using the Brier score as a performance measure. The Brier score, which is commonly used for this purpose, compares a predicted probability vector $\boldsymbol{p} = (p_1, \ldots, p_K)$ with a true class $y \in \mathcal{Y}$ in terms of the following loss:

$$L(\boldsymbol{p}, y) = \sum_{i=1}^{K} \big(p_i - [\![y = y_i]\!]\big)^2$$

For comparison, we use pairwise coupling (PC) as described in Section 2.1 with logistic regression as base learner. Additionally, we used the pairwise coupling technique proposed by Hastie and Tibshirani [11], which is also quite commonly used for this purpose (PC-HT). Finally, we include one-vs-rest logistic regression (OVR) as a common approach to multi-class classification.

Table 2. Results in terms of average Brier score (± standard deviation)

data set	#ins.	#att.	#cls.	PC	PC-HT	OVR	PELARA
iris	150	4	3	0.044±0.045	0.044±0.045	0.087±0.042	0.043±0.050
glass	214	9	6	0.439±0.013	0.434±0.018	0.442±0.050	0.432±0.043
wine	178	13	3	0.044±0.028	0.044±0.028	0.037±0.023	0.044±0.025
vowel	528	10	11	0.246±0.054	0.241±0.044	0.555±0.047	0.389±0.063
vehicle	846	18	4	0.241±0.021	0.240±0.020	0.270±0.021	0.240±0.023
segment	2310	19	7	0.060±0.018	0.070±0.015	0.134±0.016	0.068±0.012
dna	2000	180	3	0.140±0.025	0.141±0.021	0.124±0.029	0.157±0.041
pendigits	7494	16	10	0.028±0.002	0.043±0.002	0.094±0.004	0.053±0.003
poker	25010	10	10	0.566±0.002	0.566±0.002	0.567±0.002	0.565±0.002
satimage	4435	36	6	0.189±0.012	0.190±0.012	0.246±0.009	0.198±0.011
svmguide4	300	10	6	0.642±0.015	0.716±0.005	0.715±0.008	0.737±0.006
svmguide2	391	20	3	0.275±0.034	0.259±0.032	0.277±0.032	0.266±0.034
letter	15000	16	26	0.228±0.009	0.291±0.006	0.473±0.005	0.336±0.008
shuttle	43500	9	7	0.068±0.003	0.067±0.002	0.135±0.003	0.061±0.002

Table 3. Runtimes in seconds for training each fold of the data; the relative runtimes are summarized in the brackets

data set	PC	PC-HT	OVR	PELARA
iris	0.19(1.63)	0.23(2.00)	0.13(1.14)	0.12(1)
glass	2.37(1.73)	2.18(1.59)	1.75(1.28)	1.37(1)
wine	0.24(1.88)	0.35(2.70)	0.33(2.51)	0.13(1)
vowel	6.08(1.04)	6.99(1.19)	0.74(0.13)	5.86(1)
vehicle	7.37(2.45)	5.51(1.83)	6.14(2.04)	3.01(1)
segment	18.80(1.77)	14.73(1.39)	17.84(1.68)	10.63(1)
dna	161.57(0.96)	166.18(0.99)	336.30(2.00)	168.54(1)
pendigits	25.87(1.30)	39.52(1.99)	46.09(2.32)	19.91(1)
poker	10.98(0.32)	62.70(1.83)	7.30(0.21)	34.29(1)
satimage	38.52(2.24)	44.13(2.57)	10.08(0.59)	17.16(1)
svmguide4	11.23(6.72)	5.42(3.25)	2.69(1.62)	1.67(1)
svmguide2	8.58(5.55)	3.07(1.98)	3.54(2.29)	1.55(1)
letter	179.76(0.33)	264.75(0.49)	25.13(0.05)	538.56(1)
shuttle	39.16(0.63)	90.00(1.44)	22.93(0.37)	62.32(1)

The results for various data sets from the UCI repository [9] are shown in Table 2, together with some statistics of the data. These results are averages over 5 repeats of 10-fold cross validation. As can be seen, OVR is clearly outperformed by the other methods. This is confirmed by a two-tailed sign test, which reports significance at the level $\alpha = 0.05$. PC, PC-HT and PELARA are almost perfectly en par (with similar numbers of wins and losses in each pairwise comparison).

The average runtimes are shown in Tables 3. Here, PELARA performs rather well and seems to be the most efficient on average. In particular, our ranking-based approach shows clear advantages over the pairwise coupling methods (while OVR is often quite fast, too).

8 Conclusions

While the problem of multi-class probability estimation is commonly tackled by means of reduction techniques, which decompose the original problem into a set of binary problems, we have proposed an alternative method that exploits the intimate connection between probability estimation and ranking. More specifically, we take advantage of recent work on *label ranking*, which provides a natural bridge between classification and probability estimation. This connection becomes especially apparent when making use of the Plackett-Luce model, a probabilistic ranking model that links classification and ranking in a seamless manner (by modeling ranking as a sequence of classifications).

Compared to the pairwise approach, our ranking-based method appears to be more solid from a theoretical point of view, especially as it does not require any ad-hoc aggregation mechanism. The corresponding reduction of complexity also comes with improvements in terms of runtime. Regarding predictive accuracy, however, the best approaches to pairwise coupling are indeed difficult to beat, especially if the number of classes is large. A plausible explanation for this observation, which is also coherent with similar findings for pairwise classification, is the redundancy produced by the quadratic number of pairwise models. Nevertheless, our results have shown that the ranking-based alternative put forward in this paper is at least competitive to state-of-the-art pairwise coupling methods.

Due to the lack of proper benchmark data, we could not yet explore what we suppose to be the main strength of our method, namely the learning of probability models from incomplete rankings, including pairwise comparisons of the form "y_i is more likely than y_j as a class label for $\boldsymbol{x}$, but also (qualitative) comparisons involving more than two labels, such as $y_3 \succ_{\boldsymbol{x}} y_5 \succ_{\boldsymbol{x}} y_1$. Currently, we are looking for data of that kind, which, despite not having been collected systematically so far, should in principle occur quite naturally in many domains.

Acknowledgments. The authors are supported by German Research Foundation (DFG).

References

1. Bradley, R., Terry, M.: Rank analysis of incomplete block designs I. the method of paired comparisons. Biometrika 39, 324–345 (1952)
2. Buja, A., Stuetzle, W., Shen, Y.: Loss functions for binary class probability estimation: Structure and applications. Technical report, University of Pennsylvania (2005)
3. Cheng, W., Dembczyński, K., Hüllermeier, E.: Label ranking methods based on the Plackett-Luce model. In: Proc. ICML 2010, pp. 215–222 (2010)
4. Cheng, W., Hühn, J., Hüllermeier, E.: Decision tree and instance-based learning for label ranking. In: Proc. ICML 2009, pp. 161–168 (2009)
5. Clemencon, S., Lugosi, G., Vayatis, N.: Ranking and empirical minimization of U-statistics. The Annals of Statistics 36(2), 844–874 (2008)
6. Cour, T., Sapp, B., Taskar, B.: Learning from partial labels. Journal of Machine Learning Research 12, 1225–1261 (2011)

7. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning 29, 103–137 (1997)
8. Flach, P.A.: Putting Things in Order: On the Fundamental Role of Ranking in Classification and Probability Estimation. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 2–3. Springer, Heidelberg (2007)
9. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
10. Fürnkranz, J.: Round robin classification. Journal of Machine Learning Research 2, 721–747 (2003)
11. Hastie, T., Tibshirani, R.: Classification by pairwise coupling. The Annals of Statistics 26(1), 451–471 (1998)
12. Herbei, R., Wegkamp, M.: Classification with reject option. Canadian Journal of Statistics 34(4), 709–721 (2006)
13. Luce, R.: Individual Choice Behavior: A Theoretical Analysis. Wiley (1959)
14. Mallows, C.: Non-null ranking models. Biometrika 44(1), 114–130 (1957)
15. Marden, J.: Analyzing and Modeling Rank Data. CRC Press (1995)
16. Niculescu-Mizil, A., Caruana, R.: Predicting good probabilities with supervised learning. In: Proc. ICML, pp. 625–632 (2005)
17. Plackett, R.: The analysis of permutations. Applied Statistics 24(2), 193–202 (1975)
18. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in Large Margin Classifiers, pp. 61–74. MIT Press (1999)
19. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. The Journal of Machine Learning Research 5, 101–141 (2004)
20. Wellman, M.P.: Some varieties of qualitative probability. In: Proc. IPMU 1994, Paris, pp. 437–442 (1994)
21. Wu, T., Lin, C., Weng, R.: Probability estimates for multi-class classification by pairwise coupling. Journal of Machine Learning Research 5, 975–1005 (2004)
22. Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: Proc. KDD, pp. 204–213 (2001)
23. Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: Proc. KDD, pp. 694–699 (2002)
24. Zhang, T.: Statistical behavior and consistency of classification methods based on convex risk minimization. Annals of Statistics 32(1), 5–85 (2004)

Adaptive Planning for Markov Decision Processes with Uncertain Transition Models via Incremental Feature Dependency Discovery

N. Kemal Ure, Alborz Geramifard, Girish Chowdhary, and Jonathan P. How

Laboratory for Information and Decision Systems, MIT
http://www.lids.mit.edu
Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA, USA
{ure,agf,girishc,jhow}@mit.edu

Abstract. Solving large scale sequential decision making problems without prior knowledge of the state transition model is a key problem in the planning literature. One approach to tackle this problem is to learn the state transition model online using limited observed measurements. We present an adaptive function approximator (incremental Feature Dependency Discovery (iFDD)) that grows the set of features online to approximately represent the transition model. The approach leverages existing feature-dependencies to build a sparse representation of the state transition model. Theoretical analysis and numerical simulations in domains with state space sizes varying from thousands to millions are used to illustrate the benefit of using iFDD for incrementally building transition models in a planning framework.

1 Introduction

Increasing the level of autonomy for unmanned aerial vehicles (UAVs) through planning algorithms is needed to tackle real-life missions such as persistent surveillance, maintaining wireless network connectivity, and search and rescue [21,27,31]. One of the common themes amongst these missions is the presence of stochasticity, such as uncertainty in the outcome of interactions with the environment or the vehicle dynamics. One typical approach for solving these stochastic decision making problems is to cast them as Markov Decision Processes (MDPs) [23] and then use reinforcement learning (RL) [30] methods to generate policies without having to know the transition model. However, a hurdle in applying RL to real-world problems is that RL methods typically require many interactions with the world in order to find good policies. Model-based RL techniques address this sample complexity problem by fitting an approximate model to the data first and then generating simulated samples from the model [29,33]. However, this approach requires a suitable estimation model with appropriate basis functions. This paper presents a scalable transition model estimation method that can be used in a model-based RL framework for solving MDPs with state-correlated uncertainty.

For motivation, consider a robot navigating from a starting point to a goal location using GPS signals. The GPS is not fully reliable and may fail in each location with a certain probability. A succesful policy guides the robot away from locations with

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 99–115, 2012.

poor signal reception. Yet, as we will see in the numerical results, a planning-modeling scheme that assumes a uniformly distributed failure rate for the GPS can lead to poor performance.

A straightforward approach for modeling the uncertainty in such problems is to estimate the parameters of the model for every state separately (*i.e.,* use a tabular representation). However for large state spaces, this may become intractable. The main aim of this paper is to present a planning/learning technique to form a good approximation of state-dependent uncertainties with low sample complexity. To this effect, we employ an adaptive function approximation technique to estimate the uncertainties that allows flexible representations and alleviates the problem of hand-picking a set of fixed features. The representation starts with a small number of state correlated basis (features) and expands the representation in regions where model parameters are not well captured. This adaptive function approximator is based on the recently developed incremental feature dependency discovery (iFDD) algorithm [13]. By using iFDD for model parameter estimation and bootstrapping techniques for replanning, we demonstrate a substantial sample complexity reduction in learning good policies. In addition, the proposed methodology also possess asymptotic convergence properties. The applicability of the proposed method to integrated model estimation and planning is experimentally demonstrated in a gridworld problem, a block building domain, and a persistent search and track (PST) mission. Simulation results show that the representations learned by iFDD are sufficiently rich and result in a substantial reduction in the sample complexity.

2 Related Work

2.1 Robust Dynamic Programming

If the uncertain transition model is assumed to be lying on a priori known set, a robust policy can be obtained by considering the worst-case situation within this set. This approach is usually known as robust dynamic programming [15], and different methods have been developed based on how the uncertainty set is modeled and how the model is selected from the uncertainty set [22,9,4]. Although policies generated with these methods usually prevent the catastrophic effects of model-environment mismatch, they often lead to conservative solutions, since the assumed uncertainty set is usually a pessimistic estimate of the actual uncertainty representation. For example in the context of the robot navigation, this approach may assume a uniform high GPS failure rate for all states. Hence the planner does not allow the agent to explore the environment.

2.2 Adaptive Dynamic Programming (ADP)

Methods in this category start with an initial representation of the model of the system and updates this model as more data is observed from the environment [16]. After updating the model representation, a new policy is obtained by applying a dynamic programming (DP) algorithm suitable to work in real time, such as asynchronous dynamic programming [14]. A successful application of such an algorithm to PST mission in hardware setting can be found in [6], where the authors improve the policy online by estimating fuel dynamics of UAVs. Bertuccelli [5] managed to improve the performance of these algorithms by combining robust and adaptive methods, which resulted in an algorithm that estimates the uncertainty set online using the observed data. Existing ADP methods estimate a fixed set of transition parameters (which can be as large as the state space). Hence such methods are limited to low-dimensional small-scale systems.

2.3 Model Based Reinforcement Learning (MBRL)

Basic idea of MBRL [7] is similar to ADP, however development of these methods in different communities led to algorithms with different behaviors and applications. Dyna architecture [33,29] builds an approximate model of the system based on the observed data and then uses this model to generate samples. These algorithms tackle large state space by applying approximation to system dynamics, however finding a good representation often requires domain expertise.

2.4 Bayesian Non-Parametric Models

Bayesian non-parametric models (BNPM)[11] are probabilistic models that can grow their complexity in response to measured data. Thus when they are used as prior distributions over models in MBRL setting, they can alleviate the problem of finding a good representation to a certain degree since the expressiveness of the representation grows as more samples are observed. In [2], BNPMs are used for state clustering for an MRBL algorithm and [17] uses BNPMs to model a rich class of motion patterns. Although these methods offer more flexible models than fixed representations, they also tend to have higher sample complexity and they may lack asymptotic convergence guarantees. Gaussian processes and kernel filters are classes of non-parametric models that leverage the theory of reproducing kernel Hilbert spaces (see e.g. [24,19]) for regression and classification problems. The idea is to associate each measured state with a kernel so that a representation can be built online. Kernel methods need several sparsification tools to ensure that the chosen set of kernels does not become intractable as different parts of the state space are explored. Performing such sparsification online is a difficult problem, because existing methods require optimization over a set of kernels which can become computationally intensive. An incremental model expansion method based on growing hidden Markov models had been proposed by [10], however it has been verified only in learning and prediction of motion patterns.

This paper addresses the aforementioned shortcomings of existing methods by developing a model expansion method that has asymptotic convergence properties, which is sample efficient and is scalable to high dimensional large-scale problems.

3 Problem Definition

The problem of sequential decision making under uncertainty is formulated as an MDP, which is defined as the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}, \gamma \rangle$, where $\mathcal{S}$ is the discrete state space, $\mathcal{A}$ is the discrete set of actions, $\mathcal{P}^a_{ss'}$ is the state transition model that denotes the probability of transitioning to state s' when action a is applied at state s. $\mathcal{R}^a_{ss'}$ is a known reward model representing the reward for executing action a in state s and transitioning to state s'. $\gamma \in [0, 1]$ is the discount factor used to balance relative weights of current and future rewards. Only MDPs with discrete and finite state space are considered. A *policy* is a mapping $\pi : \mathcal{S} \to \mathcal{A}$ from state to action space. Together with the initial state s_0, a policy forms a trajectory $z = \{s_0, a_0, r_0, s_1, a_1, r_1, \cdots\}$. For each time-step t, $a_t = \pi(s_t)$, and both r_t and s_{t+1} are sampled from the reward and transition models correspondingly. The *value* of each state-action pair under policy π is defined as:

$$Q^{\pi}(s, a) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a\right], \tag{1}$$

which is the expected sum of discounted rewards obtained starting from state s, taking action a and following the policy π thereafter. The optimal policy π^* is given by the solution of the following optimization problem:

$$\pi^*(s) = \operatorname*{argmax}_a Q^{\pi^*}(s, a). \tag{2}$$

The optimal solution Q^* satisfies the Bellman equation

$$Q^*(s, a) = \underset{s'}{E}\left[\mathcal{R}^a_{ss'} + \max_{a'} \gamma Q^*(s', a')\right]. \tag{3}$$

3.1 MDPs with Parametric Uncertainties

An MDP with parametric uncertainty is defined by the tuple, $\mathcal{M}_p = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}(p), \mathcal{R}^a_{ss'}, p, \gamma\rangle$, where p is an unknown mapping of the form $\mathcal{S} \to [0, 1]$ and $\mathcal{P}^a_{ss'}(p)$ is the transition model as an explicit function of the unknown mapping p. The rest of elements are identical to the earlier definition of the MDP. If the mapping p is constant over all states, it corresponds to the *MDP with an unknown parameter* framework, which has been extensively studied in [5]. In a more general setting, p does not necessarily map each state to the same probability value, resulting in the *MDP with state-correlated uncertainty* framework. Solving such MDPs is the central theme of this paper.

From a practical point of view, p usually denotes occurrence probability of some event E. Here an event E refers to a set of state transitions $\langle s, a, s'\rangle, s' \sim \mathcal{P}^a_s$, that satisfy a certain condition. For instance, in the context of the robot navigation problem, an event may be defined as all state transitions where GPS signal was not received. Since the probability of the event occurrence depends on the state (*i.e.,* robot's location), then the problem needs to be formulated as an MDP with a state correlated uncertainty. In this setting, $p(s)$ can be written as the following set:

$$p(s) = P(\langle s, a, s'\rangle \in E | s). \tag{4}$$

For brevity, we only consider a single event that is action independent. For example in the robot navigation problem, the event is the GPS signal failure and it is assumed to be independent of the immediate action taken by the robot. Consequently we have removed the dependence of p on E and a in Eq 4. The extension is straight forward.

3.2 The Big Picture of Our Approach

Note that if p is known, then MDP with parametric uncertainty reduces to description of a regular MDP. Hence the underlying hypothesis of our approach is that the optimal planning problem for an MDP with parametric uncertainty can be solved by first estimating the mapping p and then solving the corresponding MDP. This is equivalent to estimating the structure of the mapping p. However the structure of many state dependencies is usually not known beforehand, as some states may be contributing significantly to the mapping $p(s)$, while others may be completely independent of it. Hence our problem definition is as follows: given an MDP with state-correlated uncertainty, develop an iterative estimation/planning scheme where parameters are estimated from the environment and a good policy is obtained with small number of total interactions with the model and the environment.

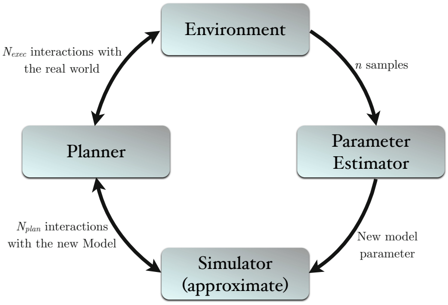


Fig. 1. Components of the adaptive planning framework

A general layout for this approach is given in Figure 1, for which one iteration of the algorithm is as follows. At the k^{th} iteration, the simulator model has an estimate $\hat{p}^k$ of the the parameter. The planner interacts with the simulator for N_{plan} steps in order to design a policy π^k. This policy is executed in the actual environment for N_{exec} steps, where it is expected that $N_{exec} \ll N_{plan}$ because collecting samples is much more expensive from the real world compared to the simulation. The resulting trajectory z^k, which is of length N_{exec}, is used by the parameter estimator to obtain the new estimate $\hat{p}^{k+1}$ with which the simulator model is updated. In the next iteration, the planner computes an improved policy π^{k+1}, and the loop continues. Based on this discussion, when the model class and policy class include the true model and the optimal policy then the optimal adaptive planning criteria can be given as: $\lim_{k\to\infty} \hat{p}^k = p$, $\lim_{k\to\infty} \pi^k = \pi^*$, meaning that the estimate converges to its true value, while the policy converges to the optimal policy. However most planning and estimation algorithms have only asymptotic convergence, requiring $N_{plan}, N_{exec} \to \infty$ to satisfy this criteria.

4 Estimation and Planning Methods

4.1 Updating the Approximate Uncertainty Representation

An approximate representation of the uncertainty can be formed using linear function approximation with binary features [8] as follows

$$p(s) \approx \hat{p}^k(s) = \phi^\top(s)\theta^k, \tag{5}$$

where $\hat{p}^k(s)$ is the approximate representation at k^{th} step and $\phi(s)$ is the vector of features [1]. Each component ϕ_j is a binary feature characterized by a mapping from state to a binary value; $\phi_j(s) : s \to \{0,1\}, j = 1, ..., m$, where m is the total number of features and $\theta^k \in \mathbb{R}^m$ is the weight vector at step k. A feature ϕ_j is called *active* at state s if $\phi_j(s) = 1$. Set of active features at state s is given by $\mathsf{A}(s) = \{j | \phi_j(s) = 1\}$. Hence, Eq. 5 can be written as:

$$\hat{p}^k(s) = \sum_{j \in \mathsf{A}(s)} \theta_j^k,$$

where θ_j^k denotes the j^{th} component of θ^k.

New estimates are formed by updating the weight vector at each step. For that purpose, define a Bernoulli random variable $\Psi(s)$ with parameter $p(s)$. That is, $\Psi(s)$ is equal to 1 with probability $p(s)$ and zero otherwise. Let z^k be the k^{th} experienced trajectory with length N_{exec}, obtained from interacting with the environment. Define $s^{k,l}$ as the state at the l^{th} time-step of the k^{th} trajectory where $l = 1, ..., N_{exec}$. Let $\theta^{k,l}$ denote the corresponding weight vector. Define $\zeta(s^{k,l})$ to be the value that random variable $\Psi(s^{k,l})$ takes. This value can be gathered from z^k based on the occurrence of the event that is defined in Eq. 4. The weight vector can be updated applying a gradient descend type update as follows

$$\begin{aligned} \theta^{k,l+1} &= \theta^{k,l} - \frac{1}{2}\alpha^{k,l}\frac{\partial}{\partial \theta^{k,l}}[p(s^{k,l}) - \hat{p}^{k,l}(s^{k,l})]^2 \\ &= \theta^{k,l} + \alpha^{k,l}[p(s^{k,l}) - \hat{p}^{k,l}(s^{k,l})]\phi(s^{k,l}), \end{aligned}$$

where $\alpha^{k,l}$ is a scalar step-size parameter. Since p is unavailable to the algorithm, it is replaced by its estimate $\zeta(s^{k,l})$. Define the sampled estimation error at state s as $\Delta p^{k,l}(s) = \zeta(s) - \hat{p}^k(s) = \zeta(s) - \phi(s)^T\theta^{k,l}$. Then, the final form of the parameter update law is

$$\theta^{k,l+1} = \theta^{k,l} + \alpha^{k,l}\Delta p^{k,l}(s^{k,l})\phi(s^{k,l}). \tag{6}$$

Eq. 6 is a variant of the well studied stochastic gradient descent (SGD) algorithm [18]. Since the structure of p is not known beforehand, quality of the resulting approximation found by SGD depends strongly on the selected set of features.

[1] Our methodology can be extended to state-action correlated uncertainties by introducing feature vectors $\phi(s,a)$.

4.2 Adaptive Function Approximation Using iFDD

Adaptive function approximators also modify the set of features based on the observed data based on the following general update rule:

$$\hat{p}^{k,l}(s) = \phi^{k,l}(s)^\top \theta^{k,l}, \qquad (7)$$
$$\phi^{k+1,l+1}(s) = h(z^k, \theta^{k,l}, \phi^{k,l}),$$

where h is the representation expansion function that adds new features to the feature vector based on sampled trajectories, weight vector, and previous set of features. Based on the successful results in representing high dimensional value functions, we employ iFDD [13,12] as the adaptive function approximator for our framework to represent the uncertainty. The basic idea of iFDD is to expand the representation by adding conjunctions of the initial features based on an error metric, thus reducing the error in parts of the state space where the feedback error persists. The general outline of the algorithm is as follows: given a set of initial binary features, when performing the update for state $s \in z^k$, a conjunction set $\phi^c(s) = \{\phi_j(s) \wedge \phi_k(s) | j, k \in \mathsf{A}(s)\}$ is formed. These features are referred as candidate features. If the sum of sampled estimation error $\Delta p(s)$ over active candidate features exceeds some pre-determined threshold ξ, these conjunctions are added to set of features. The evaluation function learner (ELF) algorithm [32], expands the representation akin to iFDD that we use, yet candidate features are selected based on a limited set of heuristically selected features.

4.3 Planning

The goal of the planning algorithm is to find a value function which satisfies the Bellman equation (*i.e.,* Eq. 3) as closely as possible. Since in our approach an approximate model is always present, we focus our attention on DP types of planners. A particular property of interest is the sample efficiency of the algorithm, meaning that a policy with reasonable expected cumulated reward is obtained with small number of interactions with the model (*i.e.,* N_{plan}). For this purpose, we compare two DP approaches: 1) classical Value Iteration (VI) [30] and 2) Trajectory Based Value Iteration (TBVI) [30], which can be viewed as an specific instance of Real Time Dynamic Programming (RTDP)[1].

Value Iteration. VI is a classic DP algorithm that updates state-action values by sweeping through the whole space, applying the *Bellman update*

$$Q(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \mathrm{max}_{a'} Q(s', a')\right], \qquad (8)$$

until no significant change is observed.[2] In our framework the number of state updates are limited by N_{plan}. When $N_{plan} \ll |\mathcal{S}|$, VI may not find reasonable approximation to the value function as Bellman updates may be applied to states with trivial change to

[2] This formula represents the asynchronous VI [3], as new estimates are used instantaneously for future estimates.

the value function. *Prioritized Sweeping* [20] addresses this problem, by applying the Bellman update to the state with the highest predicted value change. While efficient for performing exact DP, when the number of updates is fixed, this method may also focus updates in regions of the state space where the Bellman error is high, yet not frequently visited.

Trajectory Based Value Iteration. Motivated by on-policy RL methods, Trajectory Based Value Iteration (TBVI), focuses the Bellman updates in states that are sampled through Monte-Carlo simulations. The policy used for generating trajectories are ϵ-greedy with respect to the current value function:

$$\pi^{\epsilon}(s,a) = \begin{cases} 1-\epsilon & a = \text{argmax}_a Q(s,a) \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases} . \tag{9}$$

The ϵ parameter assures that in the limit all state-action pair are updated infinitely, guaranteeing the convergence to the optimal value function. Notice that both TBVI and VI apply the Bellman update (*i.e.,* Eq. 8) to N_{plan} state-action pairs. Their difference lies on their choice for selecting state-action pairs for the update. TBVI focuses the Bellman updates in regions of the state space that are deemed important based on the current policy. We will investigate the effectiveness of TBVI against VI in Section 6.4.

5 Theoretical Analysis

This section investigates the asymptotic properties of iFDD combined with the SGD algorithm presented in the previous section (iFDD-SGD). For the analysis, we consider the estimation part, assuming that the iteration number k is fixed and that each state $s^{k,l}$ and its corresponding random variable $\Psi(s^{k,l})$ is sampled by following an exploratory policy that turns the underlying MDP into an ergodic Markov chain.[3] For brevity, superscript k will be dropped from notation since it is fixed. We provide a theoretical proof showing that iFDD-SGD asymptotically converges to a solution with probability one using existing results on stochastic gradient descent theory [28,18]. Moreover, we show that if p can be captured through the representation class, iFDD-SGD converges to this point. Throughout the section we assume the standard diminishing step-size parameter for Eq. 6.

5.1 Preliminaries

Define $\mathcal{V}$ as space of all functions of the form $v : \mathcal{S} \to \mathbb{R}$. Define tabular representation as a set of features of the form $\phi_j(s_i) = \delta_{ij}$, where δ_{ij} is the Kronecker delta function and $s_i \in \mathcal{S}, i = 1, 2, ..., |\mathcal{S}|$. Note that tabular representation forms a basis for this space, since any $v \in \mathcal{V}$ can be represented as $v = \sum_{j=1,2,\ldots,|\mathcal{S}|} v\left(s_j\right) \phi_j$. It can be easily shown that ϕ_j are orthogonal to each other, hence the $\dim(\mathcal{V}) = |\mathcal{S}|$. Let $f = \{\phi_j \in$

[3] Here we are restricting ourselves to the class of MDPs that can be turned into an ergodic Markov Chain.

$\mathcal{V}, j = 1, \cdots, n\}$ be a set of n linearly independent features. Define $\Phi_f \in \mathbb{R}^{|\mathcal{S}|\times n}$ to be the feature matrix with elements $\Phi_{(i,j)} = \phi_j(s_i), i = 1, 2, ..., |\mathcal{S}|, j = 1, 2, ..., n$. It can be shown that $\mathtt{span}(\Phi_f)$ is a subspace of $\mathcal{V}$ [12], hence the orthogonal projection operator $\Pi^{\mathcal{F}} : \mathcal{V} \to \mathcal{F}$ is well defined, where $\mathcal{F}$ is the subspace $\mathtt{span}(\Phi_f)$. The weight matrix used for the projection operator is a diagonal matrix with the steady state distribution of states as its non-zero elements. More details about the matrix description of the projection operator can be found in [12]. Moreover, define $\Omega(f)$ as the set of all possible conjunctions of the elements of $\phi_j \in f$ and let $\Omega(\mathcal{F})$ be the corresponding subspace. Define $C(s)$ as the total number of non-zero features at state s. Finally define D_{ϕ_i} as the set of features constituting feature ϕ_i. For instance if $\phi_i = \phi_j \wedge \phi_k \wedge \phi_l$ then $D_{\phi_i} = \{\phi_j, \phi_k, \phi_l\}$. If ϕ_i belongs to set of initial features then, $D_{\phi_i} = \{\phi_i\}$.

5.2 Convergence of iFDD-SGD

Lemma 1. *Under the ergodicity and diminishing step-size assumptions, using SGD with fixed feature representation f, $\hat{p}_l$ defined in Eq. 7 converges to $\Pi^{\mathcal{F}}p$ as $l \to \infty$, with probability one.*

Proof. Ergodicity assumption simply turns the problem into a least-squares parameter estimation problem for p, with infinite amount of data. With the diminishing step-size assumption, it is sufficient to invoke the results of Thm 5.3.1 in [18] showing that stochastic approximation algorithm SGD will converge to least-squares solution asymptotically. That is $\hat{p}_l = \Pi^{\mathcal{F}}p$ as $l \to \infty$. □

The following lemma states that, when a new feature, which is constructed by taking conjunctions of existing features is added to the feature set, it will only be activated at the parts of the state space with more than one active feature. This conclusion will simplify the development of the convergence theorem.

Lemma 2. *Let $g \in \Omega(f)$ be added to the set of existing features by iFDD. Then $\forall s \in \mathcal{S}$ where $C(s) \leq 1$ before adding g, then $\phi_g(s) = 0$.*

Proof. If $C(s) = 0$, proof is trivial since no feature is active at state s including ϕ_g. Let $C(s) = 1$, and let ϕ_j be the corresponding active feature. if $D_{\phi_g} \subset D_{\phi_j}$, then $\phi_g(s) = 0$ due to sparse activation property of iFDD explained in subsection 4.2. Assume that $D_{\phi_g} \not\subset D_{\phi_j}$, then there exists a $\phi_i \in f$ such that $\phi_i \in D_{\phi_g}$ and $\phi_i \notin D_{\phi_j}$. Since only ϕ_j is active at s, $\phi_i(s) = 0$, which in turn implies that $\phi_g(s) = 0$. □

Theorem 1. *Under the ergodicity and diminishing step-size assumptions, $\hat{p}^l$, using SGD (Eq. 6) with iFDD representation with an initial set of features f and discovery threshold $\xi > 0$, converges to $\Pi^{\Omega(f)}p$ as $l \to \infty$ with probability one.*

Proof. Since the number of conjunctions are finite, there exists a point at time, after which the set of features is unchanged. Let us call this terminal fixed set of features $f^\dagger$ and the corresponding subspace $\mathcal{F}^\dagger$. We show that the claim holds for all possible $\mathcal{F}^\dagger$:

- ($\Pi^{\mathcal{F}^\dagger}p = p$): This means the representation is rich enough to capture the exact p vector. Lemma 1 shows that in this case $\hat{p}^l$ converges to $\Pi^{\mathcal{F}}p$ as $l \to \infty$, with probability one.

- ($\Pi^{\mathcal{F}^\dagger} p \neq p$): This means $p \notin \mathcal{F}^\dagger$. Define $\mathcal{S}^- \subseteq \mathcal{S}$ as the set of states for which $\Delta p(s) = p(s) - \hat{p}(s) \neq 0$. We argue that $\forall s \in \mathcal{S}^-, C(s) \leq 1$. Assume this is not true and let $e(s)$ denote the cumulative sampled estimation error at state s. Due to ergodicity of the underlying Markov chain, state s will be visited infinitely many times, hence after some time $e(s)$ will exceed any pre-defined threshold ξ, and since $C(s) > 1$ iFDD algorithm would expand $f^\dagger$ with the active features at state s. This contradicts that $f^\dagger$ is the terminal fixed representation.

  Now, it is sufficient to show that $\Pi^{\mathcal{F}^\dagger} p = \Pi^{\Omega(\mathcal{F})} p$. We prove this part by showing that the projection of the asymptotic residual (*i.e.,* $\delta p = p - \Pi^{\mathcal{F}^\dagger} p$), on any unexpanded feature vector (*i.e.,* $\phi_q \in \Omega(f) \setminus f^\dagger$) is null. To do so, it is sufficient to show that $\delta p^\top \phi_q = \sum_{s \in \mathcal{S}} \delta p(s) \phi_q(s) = 0$. Since $\forall s \notin \mathcal{S}^- \Rightarrow \delta p = 0$, we can write the summation as: $\sum_{s \in \mathcal{S}^-} \delta p(s) \phi_q(s)$. On the other hand, Lemma 2 showed that $\forall \phi_q \in \Omega(\mathcal{F}) \setminus \mathcal{F}^\dagger, \forall s \in \mathcal{S}^-$, if feature q is added to the representation, then $\phi_q(s) = 0$. Hence $\forall \phi_q \in \Omega(f) \setminus f^\dagger, \delta p^\top \phi_q = 0$. Therefore adding any of the remaining features to $f^\dagger$ does not help the representation to reduce the residual error further down. So as $l \to \infty, \hat{p} \to \Pi^{\mathcal{F}^\dagger} p = \Pi^{\Omega(\mathcal{F})} p$. □

Theorem 1 proves that given an initial set of features f, iFDD-SGD asymptotically converges to the best possible approximation with probability one. The analysis also provides guidance on how to select the set of initial features. If f is chosen such that $\dim(\Omega(\mathcal{F})) \geq |\mathcal{S}|$, iFDD-SGD's approximation will be exact asymptotically with probability one. For instance, consider the following process for selecting an initial set of features for an MDP with finite state space. Let s be represented by a d dimensional vector, where s_i corresponds to the i^{th} component. Hence $s = (s_1, s_2, \cdots, s_d)$. Let $\{v_i^1, v_i^2, \cdots, v_i^{n_i}\}$ denote the distinct values that s_i can take. Then initial features can be selected selected as follows:

$$f = \left[\phi_{11} \ldots \phi_{1n_1}\ \phi_{21} \ldots \phi_{2n_2} \ldots \phi_{dn_d}\right]^\top,$$
$$\phi_{ij}(s) = \begin{cases} 1 & s_i = v_i^j \\ 0 \text{ otherwise} \end{cases}, i = 1, ..., d, j = 1, ..., n_i, \tag{10}$$

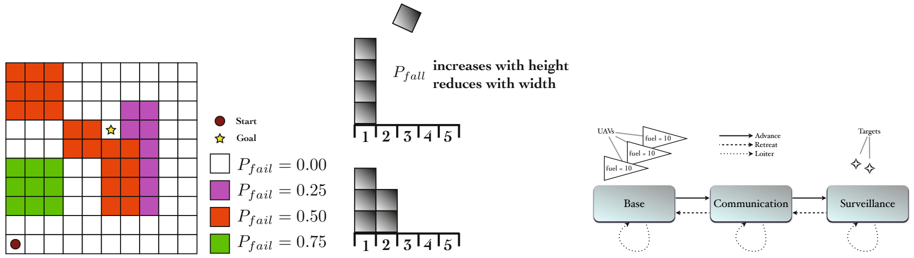
amounting to a total of $m = \sum_{i=1}^{d} n_i$ features. Geramifard et al. [12] demonstrated that, for such an initialization, $\Omega(f)$ satisfies $\dim(\Omega(\mathcal{F})) \geq |\mathcal{S}|$.

5.3 Time Complexity

The per-time-step complexity of the iFDD algorithm has been previously investigated in [12]. The study showed that, at each step given n features with maximum κ number of non-zero features, the total complexity of the feature expansion and evaluation is $\mathcal{O}(\kappa 2^\kappa)$ which is independent of the total number of features n. This property is highly desirable since n may grow to large numbers due to feature expansion, but in turn κ gets smaller due to the sparsity property of iFDD [12].

6 Simulation Study

The planning and estimation algorithms described in Section 4 are investigated in three distinct domains. The first two domains are classical RL problems: 1) gridworld and 2) block building problems motivated by [26]. Both domain descriptions are extended to include state correlated uncertainty. The third domain is a PST mission with state-correlated uncertainty in fuel dynamics. Structure of this domain is more complex and is included to validate our approach on a large-scale stochastic UAV mission planning problem. In all domains, γ was set to 0.9 and the threshold for iFDD (ξ) was set to 1.



(a) Gridworld (b) Block Building (c) Persistent Search and Track

Fig. 2. Three domains used to generate empirical results. Refer to the text for the description of each domain.

6.1 Gridworld Navigation with Sensor Management

This domain consists of a robot navigating in a 10×10 gridworld shown in Figure 2(a). The task for the robot is to reach the goal ($\star$) from the starting position ($\bullet$). Possible actions are $\{\leftarrow, \rightarrow, \uparrow, \downarrow\}$. There is 20% chance that the robot moves to one of the adjacent grids that was not intended. Reward is $+10$ for reaching the goal and zero for all movement actions. In addition, the robot carries two sensors for navigation: a camera and a GPS. The GPS is the preferred tool for navigation with no additional cost, although the probability of receiving the GPS signal is location dependent shown in Figure 2(a) as p_{fail} highlighted with four colors. If the GPS signal is not received, the robot uses the camera, incurring an additional -1 reward, while receiving the exact position. The goal of the adaptive planner is to estimate the structure of the p_{fail} through interactions and modify the plan accordingly. The size of the state-action space of the domain is about 800.

6.2 Block Building

In this domain, the objective is to distribute 5 blocks to 5 empty slots on a table shown in Figure 2(b). Initially all blocks are located under the table. The set of possible actions is defined by picking any of the blocks on or under the table and put it in any of the 5 slots. The episode is finished when there is no blocks under the table. The reward at the end of an episode is equal to the hight of the tallest tower minus one. However, placing a block on the top of the others involves p_{fall} probability of dropping the block under the table, which is increased by the size of the tower (*i.e.,* the longer the tower is, the harder is to place another block on top of it) and decreased by the number of blocks in adjacent slots. Let n_{slot} be the number of blocks in the destination slot, and n_{adj} be the maximum number of blocks in adjacent towers. Define $\bar{p}_{\text{fall}} = 0.1 \times (n_{slot} - n_{adj})^2 + 0.1$. Then p_{fall} is calculated as:

$$p_{\text{fall}} = \begin{cases} 0 & \bar{p}_{\text{fall}} \leqslant 0 \text{ or } n_{slot} = 0 \\ \bar{p}_{\text{fall}} & 0 < \bar{p}_{\text{fall}} < 1 \\ 1 & \bar{p}_{\text{fall}} \geqslant 1 \end{cases} .$$

The optimal solution is shown in the bottom of Figure 2(b). The state-action size for this domain is approximately 15×10^3.

6.3 Persistent Search and Track Mission

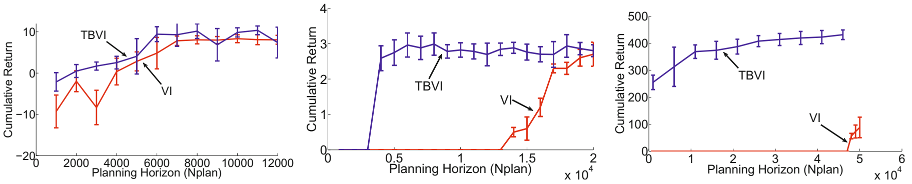
The persistent search and track is a multi-agent UAV mission planning problem, where a group of UAVs perform surveillance on a group of targets, while maintaining communication and health constraints [6]. Outline of the mission is displayed in Figure 2(c). It should be emphasized that, although this is a multi-agent domain, decision making process is completely centralized and the state-action space consists of combination of all possible states-action pairs of each UAV. Each UAV's individual state is given by its location, fuel, and health. The health is defined by two bits indicating the functionality of the sensor and the actruator. There are three available actions for each UAV: $\{$Advance, Retreat, Loiter$\}$. The objective of the mission is to travel to the surveillance node and put surveillance on two targets, while one UAV stays at communication to provide the data link with the base. Each UAV starts with 10 units of fuel and burns one unit for all actions with probability p_{nom} and 2 units with probability $1 - p_{\text{nom}}$. Since UAVs are subject to more aggressive maneuvers in the surveillance area, p_{nom} should decrease in that region. Similarly when a UAVs health is degraded, maneuverability and fuel consumption degrade accordingly. Therefore p_{nom} is a state correlated uncertainty shown in the Table 1. If a UAV runs out of fuel, it crashes and can no longer continue the mission. UAVs are also subject to sensor and actuator failures at each transition step with probability $p_{\text{fail}} \in [0, 1]$. A UAV with failed sensor cannot perform surveillance whereas a UAV with failed actuator cannot perform neither surveillance nor communication. When a UAV returns to the base, it is refueled and its failures are repaired. Hence the objective of the planner is to maximize the number of time steps that two UAVs with working sensors are located in the surveillance region and having one UAV with a working actuator in the communication region. The complete MDP description of the mission can be found in [25]. This domain has approximately 19×10^6 state-action pairs.

6.4 Results

First, we evaluated VI and TBVI across all the three domains using their exact models. The exploration schedule (ϵ) for the TBVI followed the form: $\epsilon^k = \frac{0.9}{(\text{Episode number})^{\epsilon_{\text{decay}}}} +$

Table 1. Probability of nominal fuel burn for UAVs across different locations and health status

	Health Status		
Location	No Failure	Sensor Failed	Actuator Failed
Base	1.0	1.0	1.0
Communication	0.95	0.92	0.86
Surveillance	0.88	0.80	0.75



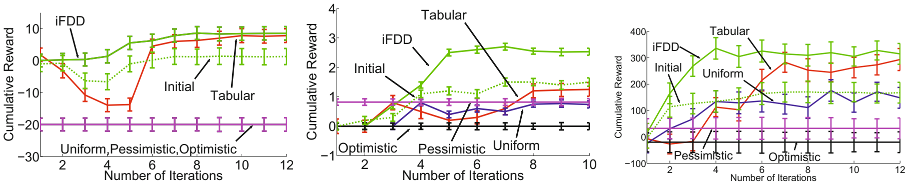
(a) Gridworld (b) Block Building (c) Persistent Search and Track

Fig. 3. Comparison of VI and TBVI across experiment domains using their exact model of the domain. The X-axis is the number of Bellman update, while the Y-axis represents the performance of the resulting policy.

0.1, where k is the planning step and ϵ_{decay} was empirically selected out of $\{0.01, 0.1, 0.5, 1.0\}$. Figure 3 depicts the results of running TBVI and VI in all three domains. The X-axis represents the number of Bellman updates, and the Y-axis corresponds to the performance of the resulting greedy policy with respect to the estimated value function. Each point shows the mean and the standard error representing 95% confidence interval based on 30 runs. Overall, our empirical results coincide with the earlier experiments [30], showing the sample efficiency of TBVI against VI. Notice as the size of the state space grew, the sample complexity reduction of TBVI over VI became more evident. In particular, in the largest domain, VI with 5 times more data compared to TBVI, achieved less than half of the TBVI's performance. This led us to pick TBVI as our planner.

Secondly, we tested the performance of model estimation methods with fixed N_{plan} and N_{exec} combined with TBVI. N_{plan} was selected based on the performance of the planners in Figure 3, while N_{exec} was chosen based on the domain properties and to be a substantially smaller number than N_{plan}. Figure 4 plots the results. The X-axis is the number of iterations. Each iteration is a complete cycle of the process displayed in Figure 1. Each algorithm was executed 30 times and for each execution, after completion of each iteration the resulting policy was evaluated on the actual domain over 30 runs amounting to 900 samples per data point. The mean of the cumulated reward and standard deviation is plotted accordingly on the Y-axis.

Five different representation was used to compare various model estimation methods. In order to emphasize the value of adaptation, fixed model estimators were also included in the evaluation. *Fixed Model Optimistic* and *Fixed Model Pessimistic* approaches assumed that the unknown parameter is fixed to 0 and 0.6 correspondingly across all states and did not update this value. *Uniform* representation ignored the state-correlation in the problem by utilizing a single fixed feature which was active at all



(a) Gridworld (b) Block Building (c) Persistent Search and Track

Fig. 4. Comparison of five estimation methods combined with TBVI across across experiment domains. The X-axis is the number of iterations of the process shown in Figure 1, while the Y-axis represents the performance of the resulting policies.

times and was adapted based on observations. *Tabular* representation stored a distinct feature for each state, resulting in a number of of parameters equivalent to the size of the state space. Initial features for iFDD were selected using Eq. 10. To emphasize the value of discovering new features, results with the fixed initial features were also included in the plots.

Gridworld. For adaptive planning experiment, planning and execution horizons were set to $N_{plan} = 8000$ and $N_{exec} = 100$. Performance of various estimation schemes using TBVI planner are displayed in Figure 4(a). In this case uniform estimation converged to the same policy obtained by any fixed model planners. This is due to fact that with a uniform uncertainty the optimal policy is to move directly towards the goal. This explains the poor performance of optimistic, pessimistic, and uniform estimators. Both tabular and iFDD estimators had the capability to capture the p_{fail} accurately. We conjecture that the dip on the performance of the tabular estimator is due to policy switching. Initially the agent followed the shortest route to the goal. As more samples were obtained, the agent explored the safe route from the right side, yet it required many samples to master the new route. The iFDD estimator performed substantially better early on compared to tabular estimator due to generalization of the features mentioned in section 4.2. In this experiment iFDD started with 22 features and expanded on average a total of 150 features. iFDD representation also performed better than representation that uses only initial features, which emphasizes the importance of expanding the representation.

Block Building. In this domain $N_{plan} = 6000$ and $N_{exec} = 30$. Trajectories were capped at 200 steps. Results are given in Figure 4(b). The optimistic model achieved 0 performance, because it assumed that $p_{\text{fall}} = 0$. Hence it kept trying to build a single tower which resulted in frequent collapses. The pessimistic approach placed 4 blocks into 4 slots and placed the final block on one of the placed blocks, achieving performance of 1. Uniform estimation eventually converged to the same policy as the pessimistic model. Both Tabular and iFDD estimators had the capability to capture the model exactly. While tabular estimator outperforms previous methods, iFDD estimator learned the task substantially faster and reached very close to the optimal policy shown in bottom part of Figure 2(b), using on average $\sim 10^3$ features.

Persistent Search and Track. In this domain $N_{plan} = 10^5$ and $N_{exec} = 1000$. Results are shown in Figure 4(c). Both fixed model approaches perform poorly. The optimistic model assumed no uncertainty in the fuel dynamics, which resulted in an overly aggressive policy with frequent crashes. The pessimistic model assumed high uncertainty in the fuel dynamics, which resulted in a conservative policy that called UAVs back to the base early, resulting in a poor performance. Even though uniform estimation outperformed both fixed estimators, its performance did not improve after a number of trajectories due to its inability to capture the state-correleated uncertainty. Representation with initial features outperformed uniform, optimistic and pessimistic approaches, yet was not competitive. A similar trend to results presented before was observed between Tabular and iFDD estimators – the iFDD estimator settled around the best found accumulated reward among all methods much faster then the tabular estimator due to its capability to represent the uncertainty with fewer parameters. In particular, iFDD was ~ 2.5 times more sample efficient than tabular estimation according to Figure 4(c). In this experiment, iFDD expanded a total of $\sim 10^4$ features on average. The size of the parameter for the tabular representation was equivalent to the size of the state space which was larger by about 70 times.

7 Conclusion

The problem of planning for MDPs with unknown state-correlated uncertainties was considered. Incremental feature dependency discovery (iFDD) was employed as a compact estimator of the state correlated uncertainty together with trajectory based value iteration (TBVI) as the planner. We proved that with a fixed policy and any set of initial features our iFDD-SGD approach will converge to the best approximated model with probability one. In particular, when the true model lies in the space of the fully expanded features, the approximation becomes exact asymptotically. The performance of the resulting algorithm was evaluated over three domains and compared against five uncertainty representations. Numerical experiment results highlighted a statistically significant improvement in terms of sample complexity and performance.

References

[1] Singh, S.P., Barto, A.G., Bradtke, S.J.: Learning to act using real-time dynamic programming. Artificial Intelligience 72(1-2), 81–138 (1995)

[2] Asmuth, J., Li, L., Littman, M., Nouri, A., Wingate, D.: A Bayesian sampling approach to exploration in reinforcement learning. In: Proceedings of the Proceedings of the Twenty-Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI 2009), pp. 19–26. AUAI Press, Corvallis (2009)

[3] Bertsekas, D.P.: Dynamic Programming and Optimal Control, 3rd edn., vol. I-II. Athena Scientific, Belmont (2007)

[4] Bertuccelli, L.F., How, J.P.: Robust Markov decision processes using sigma point sampling. In: American Control Conference (ACC), June 11-13, pp. 5003–5008 (2008)

[5] Bertuccelli, L.F.: Robust Decision-Making with Model Uncertainty in Aerospace Systems. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA (September 2008)

[6] Bethke, B., How, J.P., Vian, J.: Multi-UAV Persistent Surveillance With Communication Constraints and Health Management. In: AIAA Guidance, Navigation, and Control Conference (GNC) (August 2009) (AIAA 2009-5654)

[7] Brafman, R.I., Tennenholtz, M.: R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. Journal of Machine Learning Research (JMLR) 3, 213–231 (2001)

[8] Busoniu, L., Babuska, R., Schutter, B.D., Ernst, D.: Reinforcement Learning and Dynamic Programming Using Function Approximators. CRC Press (2010)

[9] Delage, E., Mannor, S.: Percentile Optimization for Markov Decision Processes with Parameter Uncertainty. Subm. to Operations Research (2007)

[10] Vasquez, T.F.D., Laugier, C.: Incremental Learning of Statistical Motion Patterns With Growing Hidden Markov Models. IEEE Transcations on Intelligent Transportation Systems 10(3) (2009)

[11] Fox, E.B.: Bayesian Nonparametric Learning of Complex Dynamical Phenomena. PhD thesis, Massachusetts Institute of Technology, Cambridge MA (December 2009)

[12] Geramifard, A.: Practical Reinforcement Learning Using Representation Learning and Safe Exploration for Large Scale Markov Decision Processes. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics (February 2012)

[13] Geramifard, A., Doshi, F., Redding, J., Roy, N., How, J.: Online discovery of feature dependencies. In: Getoor, L., Scheffer, T. (eds.) International Conference on Machine Learning (ICML), pp. 881–888. ACM, New York (2011)

[14] Gullapalli, V., Barto, A.: Convergence of Indirect Adaptive Asynchronous Value Iteration Algorithms. In: Neural Information Processing Systems, NIPS (1994)

[15] Iyengar, G.: Robust Dynamic Programming. Math. Oper. Res. 30(2), 257–280 (2005)

[16] Jilkov, V., Li, X.: Online Bayesian Estimation of Transition Probabilities for Markovian Jump Systems. IEEE Trans. on Signal Processing 52(6) (2004)

[17] Joseph, J., Doshi-Velez, F., Huang, A.S., Roy, N.: A Bayesian nonparametric approach to modeling motion patterns. Autonomous Robots 31(4), 383–400 (2011)

[18] Kushner, H.J., George Yin, G.: Convergence of indirect adaptive asynchronous value iteration algorithms. Springer (2003)

[19] Liu, W., Pokharel, P.P., Principe, J.C.: The kernel least-mean-square algorithm. IEEE Transactions on Signal Processing 56(2), 543–554 (2008)

[20] Moore, A.W., Atkeson, C.G.: Prioritized sweeping: Reinforcement learning with less data and less time. Machine Learning 13, 103–130 (1993)

[21] Nigam, N., Kroo, I.: Persistent surveillance using multiple unmanned air vehicles. In: 2008 IEEE Aerospace Conference, pp. 1–14 (March 2008)

[22] Nilim, A., El Ghaoui, L.: Robust Solutions to Markov Decision Problems with Uncertain Transition Matrices. Operations Research 53(5) (2005)

[23] Puterman, M.L.: Markov Decision Processes: Stochastic Dynamic Programming. John Wiley and Sons (1994)

[24] Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)

[25] Redding, J.D., Toksoz, T., Kemal Ure, N., Geramifard, A., How, J.P., Vavrina, M., Vian, J.: Persistent distributed multi-agent missions with automated battery management. In: AIAA Guidance, Navigation, and Control Conference (GNC) (August 2011) (AIAA-2011-6480)

[26] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice-Hall, Englewood Cliffs (2003)

[27] Ryan, A., Hedrick, J.K.: A mode-switching path planner for uav-assisted search and rescue. In: 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference, CDC-ECC 2005, pp. 1471–1476 (December 2005)

[28] Shapiro, A., Wardi, Y.: Convergence Analysis of Gradient Descent Stochastic Algorithms. Journal of Optimization Theory and Applications 91(2), 439–454 (1996)

[29] Sutton, R.S., Szepesvari, C., Geramifard, A., Bowling, M.: Dyna-style planning with linear function approximation and prioritized sweeping. In: Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence, pp. 528–536 (2008)

[30] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)

[31] Tozer, T.C., Grace, D.: High-altitude platforms for wireless communications. Electronics Communication Engineering Journal 13(3), 127–137 (2001)

[32] Paul, E.: Constructive function approximation. In: Feature extraction, construction, and selection: A data-mining perspective (1998)

[33] Yao, H., Sutton, R.S., Bhatnagar, S., Dongcui, D., Szepesvári, C.: Multi-step dyna planning for policy evaluation and control. In: Bengio, Y., Schuurmans, D., Lafferty, J.D., Williams, C.K.I., Culotta, A. (eds.) NIPS, pp. 2187–2195. Curran Associates, Inc. (2009)

APRIL: Active Preference Learning-Based Reinforcement Learning

Riad Akrour, Marc Schoenauer, and Michèle Sebag

TAO, CNRS − INRIA − LRI
Université Paris-Sud, F-91405 Orsay Cedex
FirstName.Name@lri.fr

Abstract. This paper focuses on reinforcement learning (RL) with limited prior knowledge. In the domain of swarm robotics for instance, the expert can hardly design a reward function or demonstrate the target behavior, forbidding the use of both standard RL and inverse reinforcement learning. Although with a limited expertise, the human expert is still often able to emit preferences and rank the agent demonstrations. Earlier work has presented an iterative preference-based RL framework: expert preferences are exploited to learn an approximate policy return, thus enabling the agent to achieve direct policy search. Iteratively, the agent selects a new candidate policy and demonstrates it; the expert ranks the new demonstration comparatively to the previous best one; the expert's ranking feedback enables the agent to refine the approximate policy return, and the process is iterated.

In this paper, preference-based reinforcement learning is combined with active ranking in order to decrease the number of ranking queries to the expert needed to yield a satisfactory policy. Experiments on the mountain car and the cancer treatment testbeds witness that a couple of dozen rankings enable to learn a competent policy.

Keywords: reinforcement learning, preference learning, interactive optimization, robotics.

1 Introduction

Reinforcement learning (RL) [26,27] raises a main issue, that of the prior knowledge needed to efficiently converge toward a (nearly) optimal policy. Prior knowledge can be conveyed through the smart design of the state and action space, addressing the limited scalability of RL algorithms. The human expert can directly demonstrate some optimal or nearly-optimal behavior, speeding up the acquisition of an appropriate reward function and/or the exploration of the RL search space through inverse reinforcement learning [23], learning by imitation [5], or learning by demonstration [19]. The use of preference learning, allegedly less demanding for the expert than inverse reinforcement learning, has also been investigated in RL, respectively to learn a reward function [6] or a policy return function [2]. In the latter approach, referred to as preference-based policy learning and motivated by swarm robotics, the expert is unable to design a reward

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 116–131, 2012.

function or demonstrate an appropriate behavior; the expert is more a knowledgeable person, only able to judge and rank the behaviors demonstrated by the learning agent. Like inverse reinforcement learning, preference-based policy learning learns a policy return; but demonstrations only rely on the learning agent, while the expert provides feedback by emitting preferences and ranking the demonstrated behaviors (section 2).

Resuming the preference-based policy learning (Ppl) approach [2], the contribution of the present paper is to extend Ppl along the lines of active learning, in order to minimize the number of expert's ranking feedbacks needed to learn a satisfactory policy. However our primary goal is to learn a competent policy; learning an accurate policy return is but a means to learn an accurate policy. More than active learning *per se*, our goal thus relates to interactive optimization [4] and online recommendation [30]. The Bayesian settings used in these related works (section 2.4) will inspire the proposed *Active Preference-based Reinforcement Learning* (April) algorithm.

The difficulty is twofold. Firstly, the above Bayesian approaches hardly scale up to large-dimensional continuous spaces. Secondly, the Ppl setting requires one to consider two different search spaces. Basically, RL is a search problem on the policy space $\mathcal{X}$, mappings of the state space on the action space. However, the literature underlines that complex policies can hardly be expressed in the state × action space for tractability reasons [21]. A thoroughly investigated alternative is to use parametric representations (see e.g. [25] among many others), for instance using the weight vectors of a neural net as policy search space $\mathcal{X}$ ($\mathcal{X} \subset \mathbb{R}^d$, with d in the order of thousands). Unfortunately earlier experiments suggest that parametric policy representations might be ill-suited to learn a preference-based policy return [2]. The failure to learn an accurate preference-based policy return on the parametric space is explained as the expert's preferences essentially relate to the policy behavior, on the one hand, and the policy behavior depends in a highly non-smooth way on its parametric description on the other hand. Indeed, small modifications of a neural weight vector $\mathbf{x}$ can entail arbitrarily large differences in the behavior of policy π_x, depending on the robot environment (the tendency to turn right or left in front of an obstacle might have far fetched impact on the overall robot behavior).

The Ppl framework thus requires one to simultaneously consider the parametric representation of policies (the primary search space) and the behavioral representation of policies (where the policy return, a.k.a. objective to be optimized, can be learned accurately). The distinction between the parametric and the behavioral spaces is reminiscent of the distinction between the input and the feature spaces, at the core of the celebrated kernel trick [7]. Contrasting with the kernel framework however, the mapping Φ (mapping the parametric representation $\mathbf{x}$ of policy π_x onto the behavioral description $\Phi(\mathbf{x})$ of policy π_x) is non-smooth[1]. In order for Ppl to apply the abovementioned Bayesian approaches used in interactive optimization [4] or online recommendation [30]

[1] Interestingly, policy gradient methods face the same difficulties, and the guarantees they provide rely on the assumption of a smooth Φ mapping [25].

where the objective function is defined on the search space, one should thus solve the inverse parametric-to-behavioral mapping problem and compute Φ^{-1}. However, computing such inverse mappings is notoriously difficult in general [28]; it is even more so in the RL setting as it boils down to inverting the generative model.

The technical contribution of the paper, at the core of the April algorithm, is to propose a tractable approximation of the Bayesian setting used in [4,30], consistent with the parametric-to-behavioral mapping. The robustness of the proposed approximate active ranking criterion is first assessed on an artificial problem. Its integration within April is thereafter studied and a proof of concept of April is given on the classical mountain car problem, and the cancer treatment testbed first introduced by [32].

This paper is organized as follows. Section 2 briefly presents Ppl for self-containedness and discusses work related to preference-based reinforcement learning and active preference learning. Section 3 gives an overview of April. Section 4.2 is devoted to the empirical validation of the approach and the paper concludes with some perspectives for further research.

2 State of the Art

This section briefly introduces the notations used throughout the paper, assuming the reader's familiarity with reinforcement learning and referring to [26] for a comprehensive presentation. Preference-based policy learning, first presented in [2], is thereafter described for the sake of self-containedness, and discussed with respect to inverse reinforcement learning [1,18] and preference-based value learning [6]. Lastly, the section introduces related work in active ranking, specifically in interactive optimization and online recommendation.

2.1 Formal Background

Reinforcement learning classically considers a Markov decision process framework $(\mathcal{S}, \mathcal{A}, p, r, \gamma, q)$, where $\mathcal{S}$ and $\mathcal{A}$ respectively denote the state and the action spaces, p is the transition model ($p(s, a, s')$ being the probability of being in state s' after selecting action a in state s), $r : \mathcal{S} \mapsto \mathbb{R}$ is a bounded reward function, $0 < \gamma < 1$ is a discount factor, and $q : \mathcal{S} \mapsto [0, 1]$ is the initial state probability distribution. To each policy π ($\pi(s, a)$ being the probability of selecting action a in state s), is associated policy return $J(\pi)$, the expected discounted reward collected by π over time:

$$J(\pi) = \mathbb{E}_{\pi,p,s\sim q}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h) \mid s_0 = s\right]$$

RL aims at finding optimal policy $\pi^* = \arg\max J(\pi)$. Most RL approaches, including the famed value and policy iteration algorithms, rely on the fact that a value function $V_\pi : \mathcal{S} \mapsto \mathbb{R}$ can be defined from any policy π, and that a policy $\mathcal{G}(V)$ can be greedily defined from any value function V:

$$V_\pi(s) = r(s) + \gamma \sum_a \pi(s,a) p(s,a,s') V_\pi(s') \tag{1}$$

$$\mathcal{G}(V)(s) = \text{ arg max } \{V(s')p(s,a,s'), a \in \mathcal{A}, s' \in \mathcal{S}\} \tag{2}$$

Value and policy iteration algorithms, alternatively updating the value function and the policy (Eqs. (1) and (2)), provide convergence guarantees toward the optimal policy provided that the state and action spaces are visited infinitely many times [26]. Another RL approach, referred to as direct policy learning [25], proceeds by directly optimizing some objective function a.k.a. policy return on the policy space.

2.2 Preference-Based RL

Preference-based policy learning (Ppl) was designed to achieve RL when the reward function is unknown and generative model-based approaches are hardly applicable. As mentioned, the motivating application is swarm robotics, where simulator-based approaches are discarded for tractability and accuracy reasons, and the *individual* robot reward is not known since the target behavior is defined at the collective swarm level.

Ppl is an iterative 3-step process. During the demonstration step, the robot demonstrates a policy; during the ranking step, the expert ranks the new demonstration comparatively to the previous best demonstration; during the self-training step, the robot updates its model of the expert preferences, and determines a new and hopefully better policy. *Demonstration* and *policy trajectory* or simply *trajectory* will be used interchangeably in the following.

Let $\mathcal{U}_t = \{\mathbf{u}_0, \dots \mathbf{u}_t;\ (\mathbf{u}_{i_1} \prec \mathbf{u}_{i_2}), i = 1 \dots t\}$ the archive of all demonstrations seen by the expert and all ranking constraints defined from the expert's preference up to the t-th iteration. A utility function J_t is defined on the space of trajectories as

$$J_t(\mathbf{u}) = \langle \mathbf{w}_t, \mathbf{u} \rangle$$

where weight vector $\mathbf{w}_t$ is obtained by standard preference learning, solving quadratic constrained optimization problem P [28,15]:

$$\begin{array}{rl} \text{Minimize} & F(\mathbf{w})) = \frac{1}{2}||\mathbf{w}||_2^2 + C \sum_{1 \leq i \leq t} \xi_{i_1,i_2} \\ s.t. \text{ for all } \ 1 \leq i \leq t & \langle \mathbf{w}, \mathbf{u}_{i_2} \rangle - \langle \mathbf{w}, \mathbf{u}_{i_1} \rangle \geq 1 - \xi_{i,j} \\ & \xi_{i_1,i_2} \geq 0 \end{array} \tag{P}$$

Utility J_t defines a policy return on the space of policies, naturally defined as the expectation of $J_t(\mathbf{u})$ over all trajectories generated from policy π and still noted J_t by abuse of notations:

$$J_t(\pi) = \mathbb{E}_{u \sim \pi}[\langle \mathbf{w}_t, \mathbf{u} \rangle] \tag{3}$$

In [2], the next candidate policy π_{t+1} is determined by heuristically optimizing a weighted sum of the current policy return J_t, and the diversity w.r.t. archive $\mathcal{U}_t$. A more principled active ranking criterion is at the core of the April algorithm (section 3).

2.3 Discussion

Let us discuss Ppl with respect to inverse reinforcement learning (IRL) [1,18]. IRL is provided with an informed, feature-based representation of the state space $\mathcal{S}$ (examples of such features $\phi_k(s)$ are the instant speed of the agent or whether it bumps in a pedestrian in state s). IRL exploits the expert's demonstration $\mathbf{u}^* = (s_0^*\ s_1^*\ s_2^* \ldots s_h^* \ldots)$ to iteratively learn a linear reward function $r_t(s) = \langle \mathbf{w}_t, \Phi(s) \rangle$ on the feature space. Interestingly, reward function r_t also defines a utility function J_t on trajectories: letting $\mathbf{u} = (s_0 s_1 \ldots s_h \ldots)$ be a trajectory,

$$J_t(\mathbf{u}) = \sum_{h=0}^{\infty} \gamma^h \langle \mathbf{w}_t, \Phi(s_h^*) \rangle = \langle \mathbf{w}_t, \sum_{h=0}^{\infty} \gamma^h \Phi(s_h^*) \rangle = \langle \mathbf{w}_t, \mu(\mathbf{u}) \rangle$$

where the k-th coordinate of $\mu(\mathbf{u})$ is given by $\sum_{h=0}^{\infty} \gamma^h \phi_k(s_h)$. As in Eq. (3), a policy return function on the policy space can be derived by setting $J_t(\pi)$ to the expectation of $J_t(\mathbf{u})$ over trajectories $\mathbf{u}$ generated from π.

IRL iteratively proceeds by computing optimal policy π_t from reward function r_t (using standard RL [1] or using Gibbs-sampling based exploration [18]), and refining r_t to enforce that $J_t(\pi_k) < J_t(u^*)$ for $k = 1 \ldots t$. The process is iterated until reaching the desired approximation level.

In summary, the agent iteratively learns a policy return and a candidate policy in both IRL and Ppl. The difference is threefold. Firstly, IRL starts with an optimal trajectory u^* provided by the human expert (which dominates all policies built by the agent by construction) whereas Ppl is iteratively provided with bits of information (this demonstration is/isn't better than the previous best demonstration) by the expert. Secondly, in each iteration IRL solves an RL problem using a generative model, whereas Ppl achieves direct policy learning. Thirdly, IRL is provided with an informed representation of the state space.

Let us likewise discuss Ppl w.r.t. preference-based value learning [6]. For each state s, each action a is assessed in [6] by executing the current policy until reaching a terminal state (rollout). On the basis of these rollouts, actions are ranked conditionally to s (e.g. $a <_s a'$); the authors advocate that action ranking is more flexible and robust than a supervised learning based approach [20], discriminating the best actions in the current state from the other actions. The main difference with Ppl thus is that [6] defines an order relation on the action space depending on the current state and the current policy, whereas Ppl defines an order relation on the policy space.

2.4 Interactive Optimization

During the Ppl self-training step, the agent must find a new policy, expectedly relevant w.r.t. the current objective function J_t, with the goal of finding as fast as possible a (quasi) optimal solution policy. This same goal, cast as an interactive optimization problem, has been tackled by [4] and [30] in a Bayesian setting.

In [4], the motivating application is to help the user quickly find a suitable visual rendering in an image synthesis context. The search space $\mathcal{X} = \mathbb{R}^D$ is made of the rendering parameter vectors. The system displays a candidate solution, which is ranked by the user w.r.t. the previous ones. The ranking constraints are used to learn an objective function, represented as a Gaussian process using a binomial probit regression model. The goal is to provide as quickly as possible a good solution, as opposed to, the optimal one. Accordingly, the authors use the Expected Improvement over the current best solution as optimization criterion, and they return the best vector out of a finite sample of the search space. They further note that returning the optimal solution, e.g. using the Expected Global Improvement criterion [17] with a branch-and-bound method, raises technical issues on high-dimensional search spaces.

In [30], the context is that of online recommendation systems. The system iteratively provides the user with a choice query, that is a (finite) set of solutions S, of which the user selects the one she prefers. The ranking constraints are used to learn a linear utility function J on a low dimensional search space $\mathcal{X} = \mathbb{R}^D$, with $J(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$ and $\mathbf{w}$ a vector in $\mathbb{R}^D$. Within the Bayesian setting, the uncertainty about the utility function is expressed through a belief θ defining a distribution over the space of utility functions.

Formally, the problem of (iterated) optimal choice queries is to simultaneously learn the user's utility function, and present the user with a set of good recommendations, such that she can select one with maximal expected utility. Viewed as a single-step (greedy) optimization problem, the goal thus boils down to finding a recommendation $\mathbf{x}$ with maximal expected utility $\mathbb{E}_\theta[\langle \mathbf{w}, \mathbf{x} \rangle]$. In a global optimization perspective however [30], the goal is to find a set of recommendations $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ with maximum expected *posterior utility*, defined as the expected gain in utility of the next decision. The expected utility of selection (EUS) is studied under several noise models, and the authors show that the greedy optimization of EUS provides good approximation guarantees of the optimal query.

In [29], the issue of the maximum expected value of information (EVOI) is tackled, and the authors consider the following criterion, where $\mathbf{x}^*$ is the current best solution: select $\mathbf{x}$ maximizing

$$EUS(\mathbf{x}) = \mathbb{E}_{\theta, x > x^*}[\langle \mathbf{w}, \mathbf{x} \rangle] + \mathbb{E}_{\theta, x < x^*}[\langle \mathbf{w}, \mathbf{x}^* \rangle] \tag{4}$$

Eq. (4) thus measures the expected utility of $\mathbf{x}$, distinguishing the case where $\mathbf{x}$ actually improves on $\mathbf{x}^*$ (l.h.s) and the case where $\mathbf{x}^*$ remains the best solution (r.h.s). This criterion can be understood by reference to active learning and the so-called splitting index criterion [8]. Within the realizable setting (the solution lies in the version space of all hypotheses consistent with all examples so far), an unlabeled instance $\mathbf{x}$ splits the version space into two subspaces: that of hypotheses labelling $\mathbf{x}$ as positive, and that of hypotheses labelling $\mathbf{x}$ as negative. The ideal case is when instance $\mathbf{x}$ splits the version space into two equal size subspaces; querying $\mathbf{x}$ label thus optimally prunes the version space. In the general case, the splitting index associated to $\mathbf{x}$ is the relative size of the smallest subspace: the larger, the better. In active ranking, any instance $\mathbf{x}$ likewise splits the version space into two subspaces: the challenger subspace of hypotheses ranking $\mathbf{x}$ higher than the current best instance $\mathbf{x}^*$, and its complementary subspace. In the Bayesian setting, considering an interactive optimization goal, the stress is put on the expected utility of $\mathbf{x}$ on the challenger subspace, plus the expected utility of $\mathbf{x}^*$ on the complementary subspace.

3 April Overview

Like Ppl, Active Preference-based Reinforcement Learning (April) is an iterative algorithm alternating a demonstration and a self-training phase. The only difference between Ppl and April lies in the self-training phase. This section first discusses the parametric and behavioral policy representations. It thereafter presents an approximation of the expected utility of selection criterion (AEUS) used to select the next candidate policy to be demonstrated to the expert, which overcomes the intractability of the EUS criterion (section 2.4) with regard to these two representations.

3.1 Parametric and Behavioral Policy Spaces

As mentioned, April considers two search spaces. The first one noted $\mathcal{X}$, referred to as input space or parametric space, is suitable to generate and run the policies. In the following $\mathcal{X} = \mathbb{R}^d$; policy π_x is represented by e.g. the weight vector $\mathbf{x}$ of a neural net or the parameters of a control pattern generator (CPG) [22], mapping the current sensor values onto the actuator values. As mentioned, the parametric space is ill-suited to learn a preference-based policy return, as the expert's preferences only depend on the agent behavior and the agent behavior depends in an arbitrarily non-smooth way on the parametric policy representation. Another space, noted $\Phi(\mathcal{X})$ and referred to as feature space or behavioral space, thus needs be considered. Significant efforts have been made in RL to design a feature space suitable to capture the state-reward dependency (see e.g. [11]); in IRL in particular, the feature space encapsulates an extensive prior knowledge [1]. In the considered swarm robotics framework however, comprehensive prior knowledge is not available, and the lack of generative model implies that massive data are not available either to construct an informed representation.

The proposed approach, inspired from [24], takes advantage of the fact that the agent is given for free the data stream made of its sensor and actuator values, generated along its trajectories in the environment (possibly after unsupervised dimensionality reduction). A frugal online clustering algorithm approach (e.g. ε-means [9]) is used to define sensori-motor clusters. To each such cluster, referred to as sensori-motor state (sms), is associated a feature. It thus comes naturally to describe a trajectory by the fraction of overall time it spends in every sensori-motor state[2]. Letting D denote the number of sms, each trajectory $\mathbf{u}_x$ generated from π_x thus is represented as a unit vector in $[0,1]^D$ ($||\mathbf{u}_x||_1 = 1$). The behavioral representation associated to parametric policy $\mathbf{x}$, noted $\Phi(\mathbf{x})$, finally is the distribution over $[0,1]^D$ of all trajectories $\mathbf{u}_x$ generated from policy π_x (reflecting the actuator and sensor noise, and the presence and actions of other robots in the swarm).

Note that behavioral representation $\Phi(\mathcal{X})$ does not require any domain knowledge. Moreover, it is consistent despite the fact that the agent gradually discovers its sensori-motor space; new sms are added along the learning process as new policies are considered, but the value of new sms is consistently set to 0 for earlier trajectories.

3.2 Approximate Expected Utility of Selection

Let $\mathcal{U}_t = \{\mathbf{u}_0, \ldots \mathbf{u}_{t-1}; (\mathbf{u}_{i_1} \prec \mathbf{u}_{i_2}), i = 1 \ldots t\}$ denote the archive of all demonstrations seen by the expert up the t-th iteration, and the ranking constraints defined from the expert preferences. With no loss of generality, the best demonstration in $\mathcal{U}_t$ is noted $\mathbf{u}_t$.

In Ppl the selection of the next policy to be demonstrated was based on the policy return $J_t(\pi_x) = \mathbb{E}_{u \sim \pi_x}[\langle \mathbf{w}_t, \mathbf{u} \rangle]$, with $\mathbf{w}_t$ solution of the problem (P) (section 2.2). By construction however, $\mathbf{w}_t$ is learned from the trajectories in the archive; it does not reward the discovery of new sensori-motor states (as they are associated a 0 weight by $\mathbf{w}_t$). Instead of considering the only max margin solution $\mathbf{w}_t$, the intuition is to consider the version space $\mathbf{W}_t$ of all $\mathbf{w}$ consistent[3] with the ranking constraints in the archive $\mathcal{U}_t$, along the same line as the expected utility of selection (EUS) criterion [30] (section 2.4).

The EUS criterion cannot however be applied as such, since policy return J_t and the version space refer to the behavioral, trajectory space whereas the goal is to select an element on the parametric space; furthermore, both the behavioral and the parametric spaces are continuous and high-dimensional. An approximate expected utility of selection is thus defined on the behavioral and the parametric spaces, as follows. Let $\mathbf{u}_x$ denote a trajectory generated from policy π_x. The expected utility of selection of $\mathbf{u}_x$ can be defined as in [30], as the expectation over the version space of the max between the utility of $\mathbf{u}_x$ and the utility of the previous best trajectory $\mathbf{u}_t$:

$$EUS(\mathbf{u}_x) = \mathbb{E}_{w\ in\ W_t}[max(\langle \mathbf{w}, \mathbf{u}_x \rangle, \langle \mathbf{w}, \mathbf{u}_t \rangle)]$$

Specifically, trajectory $\mathbf{u}_x$ splits version space $\mathbf{W}_t$ into a challenger version space noted $\mathbf{W}_t^+$ (including all $\mathbf{w}$ with $\langle \mathbf{w}, \mathbf{u}_x \rangle > \langle \mathbf{w}, \mathbf{u}_t \rangle$)), and its complementary subspace $\mathbf{W}_t^-$. The expected utility of selection of $\mathbf{u}_x$ thus becomes:

[2] The use of the time fraction is chosen for simplicity; one might use instead the *discounted* cumulative time spent in every sms.

[3] While we cannot assume a realizable setting, i.e. the expert's preferences are likely to be noisy as noted by [30], the number of ranking constraints is always small relatively to the number D of sensori-motor states. One can therefore assume that the version space defined from $\mathcal{U}_t$ is not empty.

$$EUS(\mathbf{u}_x) = \mathbb{E}_{w\ in\ \mathbf{W}_t^+}[\langle \mathbf{w}, \mathbf{u}_x \rangle] + \mathbb{E}_{w\ in\ \mathbf{W}_t^-}[\langle \mathbf{w}, \mathbf{u}_t \rangle]$$

The expected utility of selection of policy π_x is naturally defined as the expectation of EUS($\mathbf{u}_x$) over all trajectories $\mathbf{u}_x$ generated from policy π_x:

$$\begin{aligned} EUS(\pi_x) &= \mathbb{E}_{u_x \sim \pi_x}[EUS(\mathbf{u}_x)] \\ &= \mathbb{E}_{u_x \sim \pi_x}\left[\mathbb{E}_{w\ in\ \mathbf{W}_t^+}[\langle \mathbf{w}, \mathbf{u}_x \rangle] + \mathbb{E}_{w\ in\ \mathbf{W}_t^-}[\langle \mathbf{w}, \mathbf{u}_t \rangle]\right] \end{aligned} \quad (5)$$

Taking the expectation over all weight vectors $\mathbf{w}$ in W^+ or W^- is clearly intractable as $\mathbf{w}$ ranges in a high or medium-dimensional continuous space. Two approximations are therefore considered, defining the approximate expected utility of selection criterion (AEUS). The first one consists of approximating the center of mass of a version space by the center of the largest ball in this version space, taking inspiration from the Bayes point machine [14]. The center of mass of $\mathbf{W}_t^+$ (respectively $\mathbf{W}_t^-$) is replaced by $\mathbf{w}^+$ (resp. $\mathbf{w}^-$) the solution of problem (P) where constraint $\mathbf{u}_x > \mathbf{u}_t$ (resp. $\mathbf{u}_x < \mathbf{u}_t$) is added to the set of constraints in archive $\mathcal{U}_t$. As extensively discussed by [14], the SVM solution provides a good approximation of the Bayes point machine solution provided the dimensionality of the space is "not too high" (more about this in section 4.1).

The second approximation takes care of the fact that the two version spaces $\mathbf{W}_t^+$ and $\mathbf{W}_t^-$ are unlikely of equal probability (said otherwise, the splitting index might be arbitrarily low). In order to approximate $EUS(\mathbf{u}_x)$, one should thus further estimate the probability of $\mathbf{W}_t^+$ and $\mathbf{W}_t^-$. Along the same line, the inverse of the objective value $F(\mathbf{w}^+)$ maximized by $\mathbf{w}^+$ (section 2.2, problem P) is used to estimate the probability of $\mathbf{W}_t^+$: the higher the objective value, the smaller the margin and the probability of $\mathbf{W}_t^+$. Likewise, the inverse of the objective value $F(\mathbf{w}^-)$ maximized by $\mathbf{w}^-$ is used to estimate the probability of $\mathbf{W}_t^-$.

Finally, the approximate expected utility of selection of a policy π_x is defined as:

$$\text{AEUS}_t(\pi_x) = \mathbb{E}_{u \sim \pi_x}\left[\frac{1}{F(\mathbf{w}^+)}\langle \mathbf{w}^+, \mathbf{u}_x \rangle + \frac{1}{F(\mathbf{w}^-)}\langle \mathbf{w}^-, \mathbf{u}_t \rangle\right] \quad (6)$$

3.3 Discussion

The fact that April considers two policy representations, the parametric and the behavioral or feature space, aims at addressing the expressiveness/tractability dilemma. On the one hand, a high dimensional continuous search space is required to express competent policies. But such high-dimensional search space makes it difficult to learn a preference-based policy return from a moderate number of rankings, keeping the expert's burden within reasonable limits. On the other hand, the behavioral space does enable to learn a preference-based policy return from the little available evidence in the archive (note that the dimension of the behavioral space is controlled by April) although the behavioral description might be insufficient to describe a flexible policy.

The price to pay for dealing with both search spaces lies in the two approximations needed to transform the expected utility of selection (Eq. (5)) into a tractable criterion (Eq. (6)), replacing the two centers of mass of version spaces $\mathbf{W}_t^+$ and $\mathbf{W}_t^-$ (i.e. the solutions of the Bayes point machine [14]) with the solutions of the associated support vector machine problems, and estimating the probability of these version spaces from the objective values of the associated SVM problems.

4 Experimental Results

This section presents the experimental setting followed to validate April. Firstly, the performance of the approximate expected utility of selection (AEUS) criterion is assessed in an artificial setting. Secondly, the performance of April is assessed comparatively to inverse reinforcement learning [1] on two RL benchmark problems.

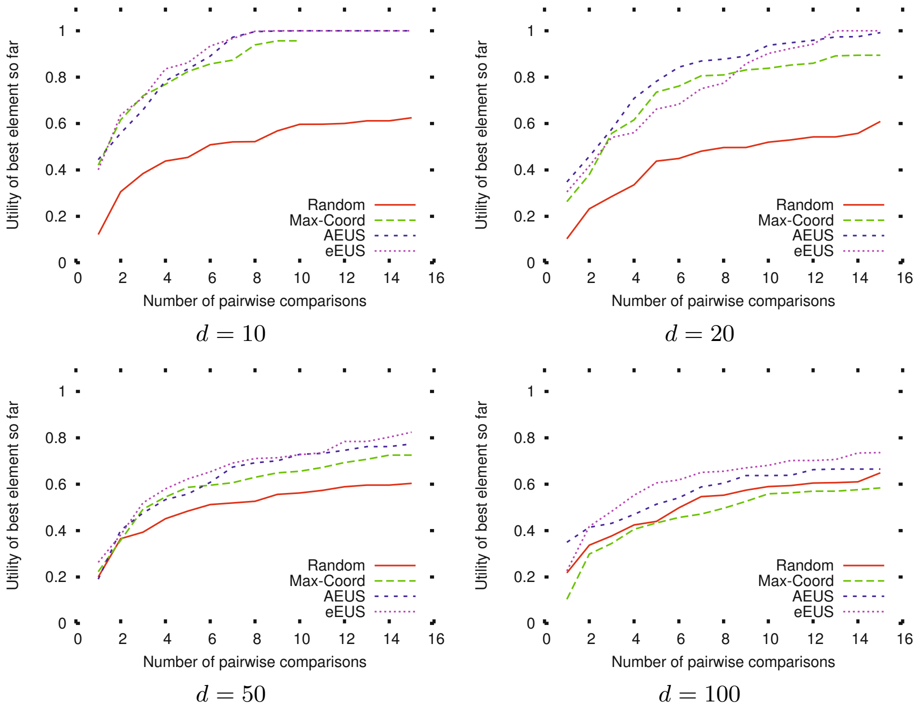
4.1 Validation of the Approximate Expected Utility of Selection

The artificial active ranking problem used to investigate AEUS robustness is inspired from the active learning frame studied by [8], varying the dimension d of the space in 10, 20, 50, 100 (Fig. 1).

In each run, a target utility function is set as a vector $\mathbf{w}^*$ uniformly selected in the d-dimensional L_2 unit sphere. A fixed sample $S = \{\mathbf{u}_1, \ldots \mathbf{u}_{1000}\}$ of 1,000 points uniformly generated[4] in the d-dimensional L_1 unit sphere is built. At iteration t, the sample $\mathbf{u}$ with best AEUS is selected in S; the expert ranks it comparatively to the previous best solution $\mathbf{u}_t$, yielding a new ranking constraint (e.g. $\mathbf{u} < \mathbf{u}_t$), and the process is iterated. The AEUS performance at iteration t is computed as the scalar product of $\mathbf{u}_t$ and $\mathbf{w}^*$.

AEUS is compared to an empirical estimate of the expected utility of selection (baseline eEUS). In eEUS, the current best sample $\mathbf{u}$ in S is selected from Eq. (5), computed from 10,000 points $\mathbf{w}$ selected in the version spaces $\mathbf{W}_t^+$ and $\mathbf{W}_t^-$ and $\mathbf{u}_t$ is built as above. Another two baselines are considered: Random, where sample $\mathbf{u}$ is uniformly selected in S, and Max-Coord, selecting with no replacement $\mathbf{u}$ in S with maximal L_∞ norm. The Max-Coord baseline was found to perform surprisingly well in the early active ranking stages, especially for small dimensions d. The empirical results reported in Fig. 1 show that AEUS is a good active ranking criterion, yielding a good approximation of EUS. The approximation degrades gracefully as dimension d increases: as noted by [14], the center of the largest ball in a convex set yields a lesser good approximation of the center of mass thereof as dimension d increases. In the meanwhile, the approximation of the center of mass degrades too as a fixed number of 10,000 points are used to estimate EUS regardless of dimension d. Random selection

[4] The samples are uniformly selected in the d-dimensional L_1 unit sphere, to account for the fact that the behavioral representation of a trajectory has L_1 norm 1 by construction (section 3.1).



$d = 10$

$d = 20$

$d = 50$

$d = 100$

Fig. 1. Performance of AEUS comparatively to baselines (see text) *vs* number of pairwise comparisons, depending on the dimension d of the search space (results averaged over 101 runs)

catches up as d increases; quite the contrary, Max-Coord becomes worse as d increases.

4.2 Validation of April

The main goal of the experiments is to comparatively assess April with respect to inverse reinforcement learning (IRL [1], section 2). Both IRL and April extract the sought policy through an iterative two-step processes. The difference is that IRL is initially provided with an expert trajectory, whereas April receives one bit of information from the expert on each trajectory it demonstrates (its ranking w.r.t. the previous best trajectory). April performance is thus measured in terms of “expert sample complexity”, i.e. the number of bits of information needed to catch up compared to IRL. April is also assessed and compared to the black-box CMA-ES optimization algorithm, used with default parameters [12].

All three IRL, April and CMA-ES algorithms are empirically evaluated on two RL benchmark problems, the well-known mountain car problem and the cancer treatment problem first introduced by [32]. None of these problems has a reward function.

Policies are implemented as 1-hidden layer neural nets with 2 input nodes and 1 output node, respectively the acceleration for the mountain car (resp. the dosage for the cancer treatment problem). The hidden layer contains 9 neurons for the mountain car (respectively 99 nodes for the cancer problem), thus the dimension of the parametric search space is 37 (resp. 397).

RankSVM is used as learning-to-rank algorithm [16], with linear kernel and $C = 100$. All reported results are averaged out of 101 independent runs.

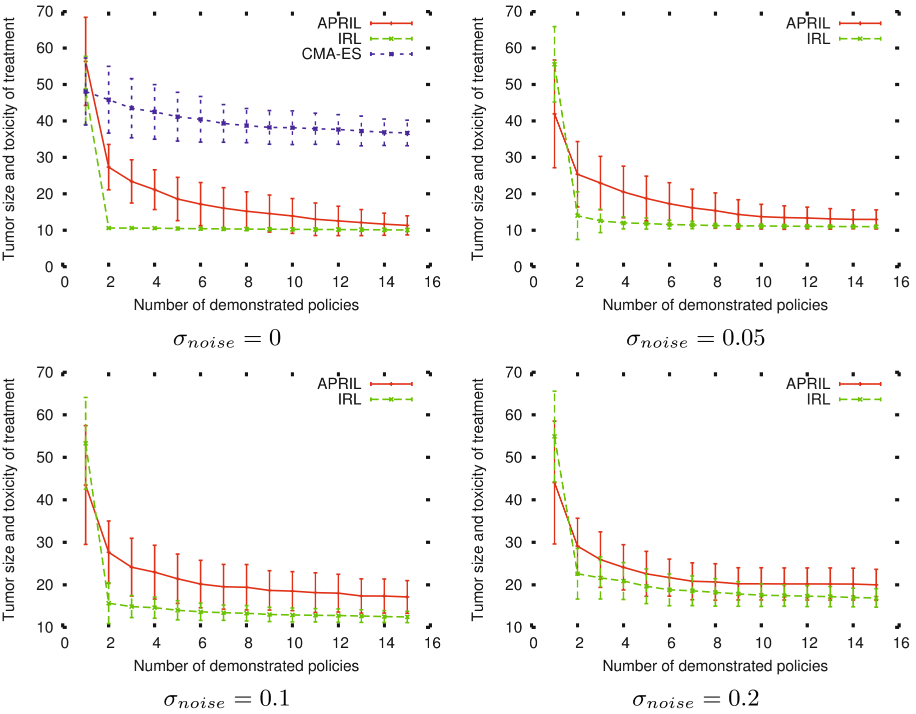
The Cancer Treatment Problem. In the cancer treatment problem, a stochastic transition function is provided, yielding the next patient state from its current state (tumor size s_t and toxicity level t_t) and the selected action (drug dosage level a_t):

$$\begin{aligned} s_{t+1} &= s_t + 0.15 \max(t_t, t_0) - 1.2(a_t - 0.5) \times 1(s_t > 0) + \varepsilon \\ t_{t+1} &= t_t + 0.1 \max(s_t, s_0) + 1.2(a_t - 0.5) + \varepsilon \end{aligned}$$

Further, the transition model involves a stochastic death mechanism (modelling the possible patient death by means of a hazard rate model). The same setting as in [6] is considered with three differences. Firstly, we considered a continuous action space (the dosage level is a real value in $[0, 1]$), whereas the action space contains 4 discrete actions in [6]. Secondly the time horizon is set to 12 instead of 6. Thirdly, a Gaussian noise ϵ with mean 0 and standard deviation σ (ranging in 0, 0.05, 0.1, 0.2) is introduced in the transition model. The AEUS of the candidate policies is computed as their empirical AEUS average over 11 trajectories (Eq. 6).

The initial state is set to 1.3 tumor size and 0 toxicity. For the sake of reproducibility the expert preferences are emulated by favoring the trajectory with minimal sum of the tumor size and toxicity level at the end of the 12-months treatment.

The average performance (sum of tumor size and toxicity level) of the best policy found in each iteration is reported in Fig. 2. It turns out that the cancer treatment problem is an easy problem for IRL, that finds the optimal policy in the second iteration. A tentative interpretation for this fact is that the target behavior extensively visits the state with zero toxicity and zero tumor size; the learned **w** thus associates a maximal weight to this state. In subsequent iterations, IRL thus favors policies reaching this state as soon as possible. April catches up after 15 iterations, whereas CMA-ES remains consistently far from reaching the target policy in the considered number of iterations when there is no noise, and yields bad results (not visible on the plot) for higher noise levels.



$\sigma_{noise} = 0$

$\sigma_{noise} = 0.05$

$\sigma_{noise} = 0.1$

$\sigma_{noise} = 0.2$

Fig. 2. The cancer treatment problem: Average performance (cumulative toxicity level and tumor size after 12-months treatment) of April, IRL and CMA-ES versus the number of trajectories demonstrated to the expert, for noise level 0, .05, .1 and .2. Results are averaged over 101 runs.

The Mountain Car. The same setting as in [31] is considered. The car state is described from its position and speed, initially set to position -0.5 with speed 0. The action space is set to $\{-1, 0, 1\}$, setting the car acceleration. For the sake of reproducibility the expert preferences are emulated by favoring the trajectory which soonest reaches the top of the mountain, or is closest to the top of the mountain at some point during the 1000 time-step trajectory.

Interestingly, the mountain car problem appears to be more difficult for IRL than the cancer treatment problem (Fig.3), which is blamed on the lack of expert features. As the trajectory is stopped when reaching the top of the mountain and this state does not appear in the trajectory description, the target reward would have negative weights on every (other) sms feature. IRL thus finds an optimal policy after 7 iterations on average. As for the cancer treatment problem, April catches up after 15 iterations, while the stochastic optimization never catches up in the considered number of iterations.

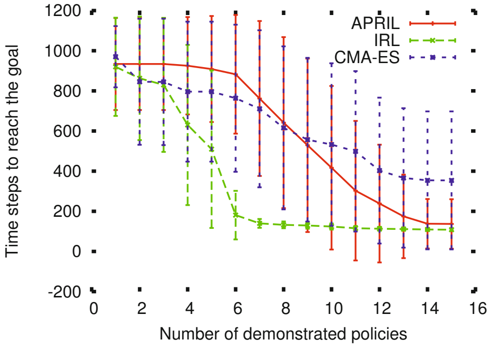


Fig. 3. The mountain car problem: Average performance (number of time steps needed to reach the top of the mountain) of APRIL, IRL and CMA-ES versus the number of trajectories demonstrated to the expert. Results are averaged over 101 runs.

5 Discussion and Perspectives

The Active Preference-based Reinforcement Learning algorithm presented in this paper combines Preference-based Policy Learning [2] with an active ranking mechanism aimed at decreasing the number of comparison requests to the expert, needed to yield a satisfactory policy.

The lesson learned from the experimental validation of APRIL is that a very limited external information might be sufficient to enable reinforcement learning: while mainstream RL requires a numerical reward to be associated to each state, while inverse reinforcement learning [1,18] requires the expert to demonstrate a sufficiently good policy, APRIL requires a couple dozen bits of information (this trajectory improves/does not improve on the former best one) to reach state of the art results.

The proposed active ranking mechanism, inspired from recent advances in the domain of preference elicitation [29], is an approximation of the Bayesian expected utility of selection criterion; on the positive side, AEUS is tractable in high-dimensional continuous search spaces; on the negative side, it lacks the approximate optimality guarantees of EUS.

A first research perspective concerns the theoretical analysis of the APRIL algorithm, specifically its convergence and robustness w.r.t. the ranking noise, and the approximation quality of the AEUS criterion. In particular, the computational effort of AEUS could be reduced with no performance loss by using Berstein races to decrease the number of empirical estimates (considered trajectories in Eq. 6) and confidently discard unpromising solutions [13].

Another research perspective is related to a more involved analysis of the expert's preferences. Typically, the expert might (dis)like a trajectory because of some fragments of it (as opposed to, the whole of it). Along this line, a multiple-instance ranking setting [3] could be used to learn preferences at the fragment (sub-behavior) level, thus making steps toward the definition of sub-behaviors and modular RL.

Another further work will be concerned with hybrid policies, combining the (NN-based) parametric policy and the model of the expert's preferences. The idea behind such hybrid policies is to provide the agent with both reactive and deliberative skills: while the action selection is achieved by the parametric policy by default, the expert's preferences might be exploited to reconsider these actions in some (discretized) sensori-motor states.

On the applicative side, April will be experimented on large-scale robotic problems, where designing good reward functions is notoriously difficult.

Acknowledgments. The first author is funded by FP7 European Project *Symbrion*, FET IP 216342, http://symbrion.eu/. This work is also partly funded by ANR Franco-Japanese project Sydinmalas ANR-08-BLAN-0178-01.

References

1. Abbeel, P., Ng, A.: Apprenticeship learning via inverse reinforcement learning. In: Brodley, C.E. (ed.) ICML. ACM International Conference Proceeding Series, vol. 69, ACM (2004)
2. Akrour, R., Schoenauer, M., Sebag, M.: Preference-based policy learning. In: Gunopulos et al. [10], pp. 12–27
3. Bergeron, C., Zaretzki, J., Breneman, C.M., Bennett, K.P.: Multiple instance ranking. In: ICML, pp. 48–55 (2008)
4. Brochu, E., de Freitas, N., Ghosh, A.: Active preference learning with discrete choice data. In: Advances in Neural Information Processing Systems, vol. 20, pp. 409–416 (2008)
5. Calinon, S., Guenter, F., Billard, A.: On Learning, Representing and Generalizing a Task in a Humanoid Robot. IEEE Transactions on Systems, Man and Cybernetics, Part B. Special Issue on Robot Learning by Observation, Demonstration and Imitation 37(2), 286–298 (2007)
6. Cheng, W., Fürnkranz, J., Hüllermeier, E., Park, S.H.: Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In: Gunopulos et al. [10], pp. 312–327
7. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
8. Dasgupta, S.: Coarse sample complexity bounds for active learning. In: Advances in Neural Information Processing Systems 18 (2005)
9. Duda, R., Hart, P.: Pattern Classification and scene analysis. John Wiley and Sons, Menlo Park (1973)
10. Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.): ECML PKDD 2011, Part I. LNCS, vol. 6911. Springer, Heidelberg (2011)
11. Hachiya, H., Sugiyama, M.: Feature Selection for Reinforcement Learning: Evaluating Implicit State-Reward Dependency via Conditional Mutual Information. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part I. LNCS, vol. 6321, pp. 474–489. Springer, Heidelberg (2010)
12. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)

13. Heidrich-Meisner, V., Igel, C.: Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In: ICML, p. 51 (2009)
14. Herbrich, R., Graepel, T., Campbell, C.: Bayes point machines. Journal of Machine Learning Research 1, 245–279 (2001)
15. Joachims, T.: A support vector method for multivariate performance measures. In: Raedt, L.D., Wrobel, S. (eds.) ICML, pp. 377–384 (2005)
16. Joachims, T.: Training linear svms in linear time. In: Eliassi-Rad, T., Ungar, L.H., Craven, M., Gunopulos, D. (eds.) KDD, pp. 217–226. ACM (2006)
17. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13(4), 455–492 (1998)
18. Kolter, J.Z., Abbeel, P., Ng, A.Y.: Hierarchical apprenticeship learning with application to quadruped locomotion. In: NIPS. MIT Press (2007)
19. Konidaris, G., Kuindersma, S., Barto, A., Grupen, R.: Constructing skill trees for reinforcement learning agents from demonstration trajectories. In: Advances in Neural Information Processing Systems, pp. 1162–1170 (2010)
20. Lagoudakis, M., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research (JMLR) 4, 1107–1149 (2003)
21. Littman, M.L., Sutton, R.S., Singh, S.: Predictive representations of state. Neural Information Processing Systems 14, 1555–1561 (2002)
22. Liu, C., Chen, Q., Wang, D.: Locomotion control of quadruped robots based on cpg-inspired workspace trajectory generation. In: Proc. ICRA, pp. 1250–1255. IEEE (2011)
23. Ng, A., Russell, S.: Algorithms for inverse reinforcement learning. In: Langley, P. (ed.) Proc. of the Seventeenth International Conference on Machine Learning (ICML 2000), pp. 663–670. Morgan Kaufmann (2000)
24. ORegan, J., Noë, A.: A sensorimotor account of vision and visual consciousness. Behavioral and Brain Sciences 24, 939–973 (2001)
25. Peters, J., Schaal, S.: Reinforcement learning of motor skills with policy gradients. Neural Networks 21(4), 682–697 (2008)
26. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
27. Szepesvári, C.: Algorithms for Reinforcement Learning. Morgan & Claypool (2010)
28. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6, 1453–1484 (2005)
29. Viappiani, P.: Monte-Carlo methods for preference learning. In: Hamadi, Y., Schoenauer, M. (eds.) Proc. Learning and Intelligent Optimization, LION 6. LNCS. Springer (to appear, 2012)
30. Viappiani, P., Boutilier, C.: Optimal Bayesian recommendation sets and myopically optimal choice query sets. In: NIPS, pp. 2352–2360 (2010)
31. Whiteson, S., Taylor, M.E., Stone, P.: Critical factors in the empirical performance of temporal difference and evolutionary methods for reinforcement learning. Journal of Autonomous Agents and Multi-Agent Systems 21(1), 1–27 (2010)
32. Zhao, K.M.R., Zeng, D.: Reinforcement learning design for cancer clinical trials. Stat. Med. (September 2009)

Autonomous Data-Driven Decision-Making in Smart Electricity Markets

Markus Peters[1], Wolfgang Ketter[1],
Maytal Saar-Tsechansky[2], and John Collins[3]

[1] Erasmus University, Rotterdam, The Netherlands
markus.peters@mphil.rsm.nl, wketter@rsm.nl
[2] University of Texas at Austin
maytal@mail.utexas.edu
[3] University of Minnesota
jcollins@cs.umn.edu

Abstract. For the vision of a Smart Grid to materialize, substantial advances in intelligent decentralized control mechanisms are required. We propose a novel class of autonomous broker agents for retail electricity trading that can operate in a wide range of Smart Electricity Markets, and that are capable of deriving long-term, profit-maximizing policies. Our brokers use Reinforcement Learning with function approximation, they can accommodate arbitrary economic signals from their environments, and they learn efficiently over the large state spaces resulting from these signals. Our design is the first that can accommodate an offline training phase so as to automatically optimize the broker for particular market conditions. We demonstrate the performance of our design in a series of experiments using real-world energy market data, and find that it outperforms previous approaches by a significant margin.

Keywords: Agents, Smart Electricity Grid, Energy Brokers, Reinforcement Learning.

1 Introduction

Liberalization efforts in electricity markets and the advent of decentralized power generation technologies are challenging the traditional ways of producing, distributing, and consuming electricity. The Smart Grid “aims to address these challenges by intelligently integrating the actions of all users connected to it . . . to efficiently deliver sustainable, economic and secure electricity supplies.” [3] This ambitious vision requires substantial advances in intelligent decentralized control mechanisms that increase economic efficiency, while keeping the physical properties of the network within tight permissible bounds [17].

A promising approach to enable the critical real-time balance between supply and demand within the network is the introduction of *electricity brokers*, intermediaries between retail customers and large-scale producers of electricity, [8]. Electricity brokers serve as information aggregators, they fulfill risk pooling and

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 132–147, 2012.

management functions, and they help attain socially desirable market outcomes given proper economic incentives. Brokers trade in multiple interrelated markets simultaneously – a structure that Bichler et al. [1] refer to as *Smart Markets*. As such, Smart Markets constitute a novel class of complex, fast-paced, data-intensive markets, in which participants employ (semi-)autonomous trading agents to attain good trading results. Importantly, because there is considerable variability in the structure that a future Smart Electricity Market might have, it is imperative that the design of an autonomous electricity broker agent can accommodate a wide variety of market structures and conditions.

We propose a novel class of autonomous Electricity Broker Agents for retail electricity trading that operate in a wide range of market structures, and that are capable of deriving long-term, profit-maximizing policies. Our brokers use Reinforcement Learning with function approximation, they can accommodate arbitrary economic signals from their environments, and they learn efficiently over the large state spaces resulting from these signals. Previous approaches are limited in the state space size they can accommodate, and are consequently constrained by the economic environments they could be deployed into. For example, previous works [11,12] do not consider customers' daily load profiles (assuming fixed consumption) and the broker's wholesale trading, both core challenges for real-world electricity brokers. We alleviate these assumptions in our simulation model. Our broker design is also the first that can accommodate an offline training phase to automatically optimize the broker for various market conditions. We demonstrate the benefits of this procedure by evaluating automatically constructed brokers for different customer populations.

The empirical evaluations we report here are based on real-world electricity market data from the Ontario Wholesale Market and industry-standard load profiles for private households. Our empirical results demonstrate that our design is effective and that it outperforms prior approaches despite the additional challenges we consider here. We hope that our broker agents contribute to current research on economic mechanism design for the Smart Grid by providing effective strategies against which such mechanisms can be validated, e.g. [16]. More generally, research on autonomous Electricity Broker Agents for the Smart Grid constitutes a nascent, emerging field, in which most of the challenges are largely unexplored. Thus, in addition to the development of a novel broker agent design, important objectives of this work are to discuss key design decisions that allow broker agents to operate effectively in the Smart Grid, and to inform future work of challenges and promising research directions.

2 Smart Electricity Market Simulation

We begin with an overview of the key entities in our Smart Electricity Market, followed by a description of the models representing them in our simulation.

Smart Electricity Markets aim to intelligently integrate the actions of *Customers*, *Generating Companies*, and the *Distribution Utility*. One promising approach to achieving this integration is introducing *Electricity Brokers* as intermediaries.

Customers are small-to-medium-size consumers and/or producers of electricity, such as private households and small firms. Customers buy and sell electricity through a *tariff market*, where electricity retailers publish standardized tariff offerings, including fixed-rate, time-of-use (ToU), and variable-rate tariffs.

Generating Companies (GenCos) are large-scale producers of energy, such as operators of fossil-fueled power plants and wind parks. GenCos are wholesalers of future electricity production commitments.

The Distribution Utility (DU) is responsible for operating the electric grid in real-time. In particular, the DU manages imbalances between the energy consumption and the total outstanding production commitments at any given time. To this end, the DU buys and sells energy on short notice and charges the responsible retailer imbalance penalties for its balancing services.

Electricity Brokers are profit-seeking intermediaries trading for their own account. They are retailers of electricity in the tariff market, and they offset the consumption of their tariff subscribers by acquiring production commitments in either the tariff market (small-scale producers) or the wholesale market (GenCos). The *portfolio* of contractual arrangements that brokers build in this way is executed in real-time by the DU. Brokers aim to build a portfolio of high-volume, high-margin tariff subscriptions with predictable consumption patterns that can be offset with production commitments at a low cost.

We developed a data-driven **Smart Electricity Market simulation** based on wholesale prices from a real-world electricity market in Ontario and electricity consumption patterns based on industry-standard load profiles. An important property of our simulation, with implications for the broker we design to operate in this environment, is to alleviate the assumption in previous works that consumers exhibit fixed demand [11,12]. Fixed demand simplifies the broker's task, however the resultant brokers may not offer an adequate response to the realities of electricity markets. In particular, a key challenge for real-world brokers is to effectively deal with *patterns* in consumer demand. This is important because some patterns (e.g. highly variable ones) are more costly for the broker to offset in the wholesale market than others.

Our simulation model is constructed from the following entities:

Electricity Broker Agents $\mathcal{B} = \{B_i\}$ or *brokers* contract with customers through the *tariff market* and procure offsetting amounts of energy in the wholesale market. Brokers publish one fixed-rate tariff at any given time. This design reflects the fact that fixed rates are currently still the dominant tariff model, mainly due to the absence of advanced metering capabilities among electricity customers. We are interested in the performance of methods for autonomous *retail electricity trading*. To this end, we endow both, our own strategies and our benchmark strategies, with a fixed wholesale trading strategy based on exponentially averaged load forecasts, and brokers learn to develop a profitable retail trading strategy against this backdrop.

Customers $\mathcal{C} = \{C_j\}$, where C_j denotes a population of customers with similar characteristics and a joint, aggregate consumption profile. We describe our customer model in more detail below. Presently, only a small proportion of electricity is produced decentrally[1] and central production will continue to play a significant role in the near future. To accommodate this design we consider customers to be exclusively consumers of electricity in this paper.

The Distribution Utility (DU) is responsible for real-time grid operation.

The Simulation Environment is responsible for coordinating brokers, customers, and the DU. It manages the tariff market, and it provides a wholesale market based on actual market data from Ontario's independent system operator (`http://www.ieso.ca`) which has also been used in a related study [12]. The wholesale market in our simulation determines prices by randomly selecting a window of appropriate size from almost ten years of real-world wholesale market pricing data. Once these prices have been determined, broker orders have no impact on them.[2]

The simulation runs over T timeslots $1, \ldots, t, \ldots, T$ which are structured as described in Figure 1:

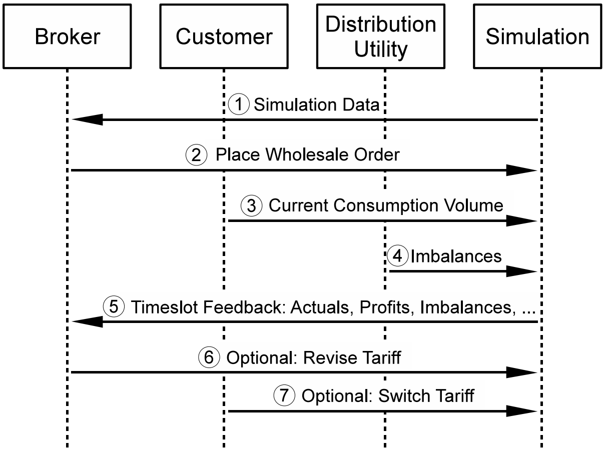

Broker
Customer
Distribution Utility
Simulation
1 Simulation Data
2 Place Wholesale Order
3 Current Consumption Volume
4 Imbalances
5 Timeslot Feedback: Actuals, Profits, Imbalances, ...
6 Optional: Revise Tariff
7 Optional: Switch Tariff

Fig. 1. Sequence diagram for one simulation timeslot

1. Each broker B_i receives information about its current customers $\mathcal{C}_t(B_i)$, the history of wholesale prices $W_1, \ldots, W_{t-1}$, the tariffs offered by all brokers

[1] As a liberal upper bound consider that, of the 592 TWh of electricity produced in Germany in 2009, merely 75 TWh were produced decentrally under the country's Renewable Energy Act (12.6%) [4].

[2] Considering brokers as price-takers is reflective of liberalized retail electricity markets, where an increasing number of small brokers compete against each other. For 2008, for example, the European Commission reported close to 940 non-main electricity retailers in Germany that shared 50% of the German market [4].

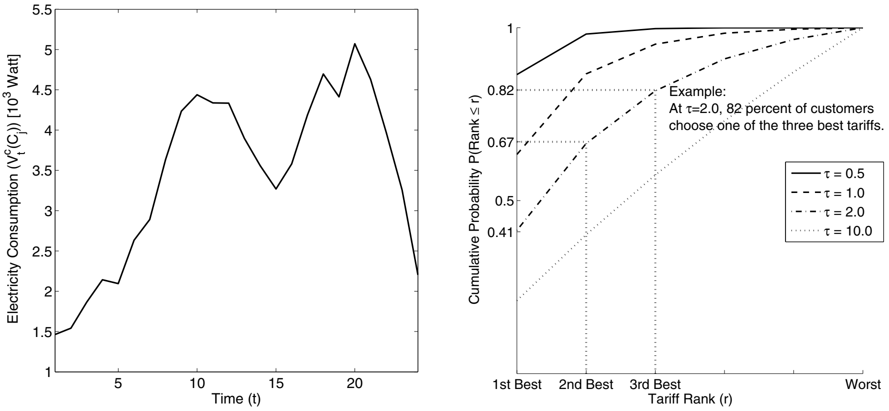
at the end of the last timeslot $\mathcal{T}_{t-1} = \{\tau_{B_1}, \ldots, \tau_{B_{|\mathcal{B}|}}\}$, and its current cash account balance.

2. Each broker indicates the volume of energy $\hat{V_t^c}$ that it wishes to procure in the current timeslot. Note, that the broker has no previous knowledge of its customers' actual consumption nor of the wholesale prices for the current timeslot. There is no acquisition uncertainty; the indicated volume $\hat{V_t^c}$ is always filled by the simulation.
3. Each customer C_j decides the volume of electricity $V_t^c(C_j)$ to consume given its current tariff, and announces this volume to the simulation. The volume consumed, $V_t^c(C_j)$, is derived from the corresponding customer's consumption model, which we describe below.
4. Based on the consumption decisions of its customers, its current tariff, and its acquisition in the wholesale market, each broker's cash account is credited (debited) with a trading profit (loss) $\tau^c(V_t^c) - \hat{V_t^c} \cdot W_t$, where $\tau^c(V_t^c)$ denotes the cost of consuming V_t^c under the current tariff τ^c to the customers (i.e. the revenue of the broker), and $\hat{V_t^c} \cdot W_t$ denotes the cost of procuring $\hat{V_t^c}$ units of energy at the prevailing wholesale price W_t. Any imbalance between the broker's forecast, and the actual amount of energy consumed by its customers is made up for by the Distribution Utility. An imbalance penalty of I per unit of mismatch, or $|V_t^c - \hat{V_t^c}| \cdot I$ in total, is debited from the cash account of the broker for this service.
5. Each broker receives ex-post information on the actual aggregate consumption volume of its customers in the current timeslot V_t^c, its trading profit, its imbalance penalty, and its cash account balance at the end of the timeslot.
6. Each broker is queried if it wishes to change its offered tariff.
7. Each customer is queried if it wishes to subscribe to a different tariff.

Customers in our simulation are represented by a customer model, each instance of which represents the aggregate behavior of *group of customers*. The customer model consists of a *consumption model*, which computes the amount of energy consumed in a given timeslot, and a *tariff evaluator*, which defines how customers select a tariff from a set of offered tariffs.[3]

The **consumption model** is based on the standard load profile (SLP) for a group of private households. SLPs are commonly used in the industry to capture characteristic load patterns under defined circumstances, e.g. [6]. To our knowledge, SLPs are the best representation available for household electricity consumption. Figure 2a shows a single day load profile generated by our consumption model. The profile reflects the characteristic consumption peaks exhibited by private households around noon and during the early evening hours.

[3] Note, that separating the consumption decision from the tariff selection decision is economically well-motivated. In the short run, the electricity demand of private households is unresponsive to changes in price level. There is some empirical evidence for customers' willingness to *shift* electricity consumption over the day in response to changing electricity prices, e.g. [7]; however, this phenomenon does not apply to our scenario of a fixed-rate tariff.



(a) One-day load profile from our consumption model.

(b) CDF for the Boltzmann distribution.

Fig. 2. Properties of our customer model

The consumption model can also be parametrized to include an arbitrary noise term around the base SLP.

Our **tariff evaluator** works as follows: If the tariff that a customer is currently subscribed to is still available, the customer considers selecting a new tariff with a fixed probability q. With probability $1 - q$ it remains in its current tariff without considering any other offers. This behavior captures customers' *inertia* in selecting and switching to new tariffs. If the tariff that the customer is currently subscribed to is not available any longer, the customer selects a new tariff with probability 1.

To select a new tariff, the customer ranks all tariffs according to their fixed rates; ties are broken randomly. A perfectly informed and rational customer would simply select the lowest-price tariff from this ranking, because the lowest-rate tariff minimizes the expected future cost of electricity. In reality, however, customer decisions will tend to deviate from this theoretical optimum for different reasons, including (1) customers do not possess perfect information about all tariffs, either because it is unavailable to them, or because they eschew the effort of comparing large numbers of tariffs; and (2) they make decisions based on non-price criteria such as trust and network effects that are absent from our model. We capture these deviations from a simple price rank-order using a Boltzmann distribution.

Assume a customer has to decide among a total of $|\mathcal{T}|$ tariffs. Then the probability of selecting the r-th best tariffs is: $Pr(\text{Rank} = r) = \frac{e^{-r/\tau}}{\sum_{i=1}^{|\mathcal{T}|} e^{-i/\tau}}$ Here, τ is the so-called *temperature* parameter with $\tau \in (0, \infty)$. The temperature can be interpreted as the customers' *degree of irrationality* relative to the theoretically optimal tariff decision. Consider the Cumulative Distribution Functions (CDF) depicted in Figure 2b for different values of τ. For $\tau \to 0$, only the best-ranked tariff has considerable mass, i.e. the tariff decision is perfectly rational. For $\tau \to \infty$ the distribution approaches a discrete uniform distribution, i.e. customers select their tariff at random.

3 Markov Decision Processes and Reinforcement Learning

To operate effectively in the Smart Electricity Market outlined in Section 2, an Electricity Broker Agent ought to learn from its environment in multiple ways. Firstly, it needs to learn about potential customers and their behavior in terms of tariff selection and electricity consumption. Secondly, it needs to learn about the behavior of its competitors and derive tariff pricing policies that strike a balance between competitiveness and profitability. And finally, it needs to learn ways of matching tariff market actions with wholesale trading strategies in order to maximize its profit. Note, that the broker's only means of learning is its ability to act in the markets it trades in, and to observe the (long-term) consequences that its actions entail.

Reinforcement Learning (RL) offers a suitable set of techniques to address these challenges, where the learner's objective is to collect the highest net present value of all present and future rewards. This could entail foregoing some immediate rewards for higher rewards in the future [13]. Numerous algorithms have been proposed for finding good policies [14]. In our scenario we use SARSA, an algorithm from the class of Temporal Difference algorithms that is well-suited for online control problems such as our retail electricity trading task. The algorithm starts out with some initial model of an action-value function $\mathcal{Q}(s, a)$, acts (approximately, except for occasional exploration) according to the policy implied by $\mathcal{Q}$, and updates $\mathcal{Q}$ with the true feedback it receives from the environment in each timeslot by $\mathcal{Q}(s, a) \leftarrow \mathcal{Q}(s, a) + \alpha[r_{t+1} + \gamma\mathcal{Q}(s_{t+1}, a_{t+1}) - \mathcal{Q}(s_t, a_t)]$ where α denotes the *learning rate*. In general, SARSA only converges to a precise estimate of $\mathcal{Q}$ when each state-action pair is visited an infinite number of times, and when the policy followed by the learner converges to a fixed policy. In our empirical evaluation we show that our learner performs well in spite of not fully meeting these theoretical requirements.

A key challenge of using RL for the problem we address here pertains to defining an effective state space. Because it is not well understood which state features are useful for capturing changes in the action-value, it is beneficial to employ a wide array of features so as to avoid the exclusion of particularly relevant ones. However, even with a limited number of features, the state space quickly becomes too large to hold in memory. Furthermore, when the state space is large, the extent of exploration required for the learner to arrive at a reliable estimate of the action values $\mathcal{Q}(s, a)$ for each $a \in \mathcal{A}$ becomes prohibitive. Previous work has dealt with this challenge by introducing *derived features* that combine multiple environmental features into a single feature for the learner [11,12]. However, these derived features are inherently less informative, and there is no principled approach to constructing them.

We alleviate these challenges by learning the broker's strategies via *function approximation*, i.e. a parametrized, functional representation of $\mathcal{Q}(s, a)$. This approach offers an attractive alternative to an explicit representation of the value in each state-action pair, thus allowing the broker to explore the effectiveness of strategies over a wider array of potentially relevant states. One type of function approximation uses the representation $\mathcal{Q}(s,a) = \boldsymbol{\theta}\boldsymbol{\mathcal{F}}(s,a)'$ where $\mathcal{Q}(s,a)$ is linear in $\boldsymbol{\mathcal{F}}(s,a)$, a vector of selected *features* of the current state s given an action a. The reinforcement learner continually updates the weights in $\boldsymbol{\theta}$ to make $\mathcal{Q}$ more representative of the experiences gathered from the environment. Other types of function approximation can be used instead of this linear scheme, e.g. [2].

4 Learning Strategies

In this section, we first introduce our function approximation based reinforcement learners, LinFA and AutoLinFA. We show how the flexible design of our learners accommodates both manual state space construction and the automatic construction of state spaces through optimization techniques. A thorough empirical evaluation of our learners in comparison to strategies proposed in the literature follows in Section 5.

4.1 Linear Function Approximation

Our first candidate strategy is **LinFA**, a reinforcement learner based on linear function approximation. In this setting, the broker uses the discrete action set shown in Table 1, which offers the broker important freedom for action:

The broker can set its tariffs **relative** to other tariffs in the market. In doing so, the broker can choose among attacking its competitors (MarginLeader), positioning itself in the middle of the market (MarginAvg), and avoiding competition altogether by posting the most expensive tariff (MarginTrailer). Alternatively, rather than setting its tariffs relative to the market, the broker can set its tariffs in an **absolute** fashion, choosing between LowMargin and HighMargin, regardless of the competing tariffs in the market. We chose the specific margins in Table 1 for their good observed performance in our experiments. Finally, the broker also has the option to leave its current tariff unchanged (NoOp).

Note, that while the brokers' ultimate action will be to set an absolute rate on its tariff, we designed the action space exclusively in terms of margins over the wholesale rate. Interestingly, we found that normalization of the rates in this manner improved the learning results drastically; otherwise, the learner can be overburdened by simultaneously learning the variability in the wholesale price-level as well as the variability among its competitors.

As state space, we first manually selected the following key features from the environment: (1) $|\mathcal{C}(B)|$ the number of customers that are currently subscribed to broker B's tariff, (2) $\mu(\tau)$ the margin of the offered tariff over the prevailing wholesale rate, and (3) $\frac{d|\mathcal{C}(B)|}{dt}$ the change in the number of customers subscribed to a broker's tariff over time. These features arguably reflect some of the most important pieces of economic information in the environment.

Table 1. Action set for LinFA and AutoLinFA

Action	**Margin over Wholesale Price**
MarginLeader	Slightly lower than cheapest competitor
MarginAvg	Average of all competitors
MarginTrailer	Slightly higher than most expensive competitor
LowMargin	Constant 10% margin
HighMargin	Constant 20% margin
NoOp	Keep the current *tariff rate*. This could lead to changes in the margin if wholesale prices change.

4.2 Offline Optimization

While LinFA's manually constructed state space is economically well-motivated, it has a number of disadvantages:

- It is unclear which other environmental features should be included in the state space, what type of feature coding should be used, and which features should be ignored for better learning performance.
- It is unclear how the set of included features depends on environmental factors such as customer characteristics, or the presence of other brokers. Certain environments might call for the inclusion of features that are otherwise distracting to the learner.
- The process of manual state space construction and validation is laborious.

Moreover, even after fixing the state space, parameters such as the learning rate α and the discount parameter γ need to be chosen manually by the user.

We aimed to address these challenges by employing heuristic optimization to identify an advantageous state space and learning parameters. Formally, let $\boldsymbol{\mathcal{F}}(s, a)$ be a set of n *candidate features* of the current state-action pair, and θ a vector of m learning parameters. Then

$$\mathcal{B}_{LinFA} = \{B_{LinFA}(\phi_1, \ldots, \phi_n, \theta_1, \ldots, \theta_m) \| \varPhi \in \{0,1\}^n, \varTheta \in \mathbb{R}^m\}$$

is a class of linear function approximation based RL brokers that use the feature $(\boldsymbol{\mathcal{F}}(s, a))_i$ as part of their state space iff $\phi_i = 1$.

To measure how well a particular broker, $B \in \mathcal{B}_{LinFA}$, competes in a particular environment, we define the *fitness function* $F : B \mapsto [0, 1]$ as the average profit share that B captures in a given number of sample simulations. The best broker B^* for the given environment is then $B(argmax_{\varPhi,\varTheta} F(B(\varPhi, \varTheta)))$.

Our second strategy, **AutoLinFA**, pertains to a class of brokers $\mathcal{B}_{LinFA}$ with the same action space as LinFA and a set of 29 *candidate features* to represent a given state-action pair. These features include the average wholesale price and the gradient of the broker's cash account. Due to space limitations, we omit the complete list of candidate features. For a given $B_{LinFA} \in \mathcal{B}_{LinFA}$, the associated fitness function F evaluates 50 simulation runs over 240 timeslots. In principle, different (heuristic) optimization methods can be used to identify effective values for Φ and Θ with respect to F. However, particularly because our parameter space consists of mostly binary features, in the experiments we report here we employed a Genetic Algorithm (GA). The results we show were produced by running the GA over 100 generations with 20 candidates per generation.

4.3 Nonlinear Function Approximation

In addition to a linear function approximation, we also explored the performance of a broker agent learnt via RL with nonlinear function approximation. Specifically, the resultant strategy, **AutoNNFA**, refers to a class of brokers $\mathcal{B}_{NNFA}$ that use Neural Networks with one hidden layer and a hyperbolic tangent sigmoid transfer function to approximate the action-value function $\mathcal{Q}$. Interestingly, in our empirical evaluations we found that AutoNNFA exhibited some desirable properties under some environmental conditions; however, its performance was not consistently superior across different environments. Specifically, AutoNNFA exhibited lower variability in performance for environments with low customer switching probabilities q and low customer irrationalities τ. For environments with higher variability and noise levels, however, AutoNNFA's performance approached that of a simple fixed-markup strategy. Its linear counterpart, AutoLinFA, competed successfully over a substantially wider range of environments. In part we attribute AutoNNFA's inconsistent performance across different environments to its slow reaction to sudden changes. In addition, in the presence of high levels of noise in the environment, we found that AutoNNFA is more likely to derive erratic policies with oscillating tariff-rates than does AutoLinFA. Because inconsistent performance is undesirable, we do not recommend its use and henceforth focus our discussion on the linear brokers.

4.4 Reference Strategies

We evaluate our Electricity Broker Agent against the table-based RL strategies proposed in [12]. To address the need for a limited state space, their strategies are learned from derived features, referred to as *PriceRangeStatus* and *PortfolioStatus*. Their simulation model does not include an explicit representation of a wholesale market, and the brokers' only sources of electricity production commitments are small-scale producers. Brokers offer one *producer tariff* in addition to the consumer tariff used by the brokers in our study. These differences make some of their results difficult to interpret in the context of the scenario we explore here.[4]

The most relevant benchmark strategies for evaluating our Electricity Broker Agent are (1) **Fixed**: a strategy which charges a constant markup μ over the smoothed wholesale price, and (2) Learning: a table-based reinforcement learner

[4] To incorporate these strategies in our simulation setting we used wholesale prices for producer prices, and suppressed actions pertaining to small-scale producer tariffs. We also excluded the state of PortfolioStatus, which is not meaningful for learning the TableRL strategy in our simulation model.

operating over the reduced, manually constructed state space outlined above. For clarify, henceforth we refer to the Learning strategy as **TableRL**. We refer the reader to [12] for complete details on these strategies.

5 Experimental Evaluation

We evaluated LinFA, our reinforcement learner with a manually constructed state space, and different automatically constructed AutoLinFA learners against the benchmark strategies from Section 4.4 in a series of experiments.

Each experiment ran over 30 simulated days (720 timeslots), in which the performance of each individual broker is computed as the share of the overall profits they captured. In the experiments we report below, the customer population is fixed to five instances of our customer model, each representing the aggregate behavior of a *group* of households.[5] The so-called *markup* parameter [12] of the reference strategies Fixed and TableRL was set to 0.05, at which we found that these strategies performed best.

5.1 Function Approximation

Figure 3 shows the performance of one **LinFA** broker in competitions against one Fixed and one TableRL broker for different customer switching probabilities q (left panel), and different levels of customer irrationality τ (right panel). LinFA is highly successful in many of these environments, and it beats both reference strategies by a statistically significant margin in all cases except for $\tau \geq 2.0$.[6] It is interesting to note that TableRL's performance lags not only behind LinFA, but also behind the Fixed strategy. This does not contradict the good performance results reported in [12], as our implementation contains only parts of their original state space (see Section 4.4). But it shows the sensitivity of RL results to a well-chosen state space, and the need for a broker design that is flexible enough to accommodate the best state space for a given environment.

For high levels of customer irrationality, the performance of LinFA approaches that of the Fixed strategy. This result may seem counter-intuitive, because even for the limiting case of customers choosing their tariffs at random, there is a

[5] We found that a larger numbers of customer groups had no significant impact on the results as they did not change the diversity of the population, while with fewer customer groups the simulation produced an unrealistic "winner takes it all" competition. Each customer model instance was parametrized with the same switching probability q and degree of irrationality τ as indicated in the figures, and noise of $\sigma = 5\%$ around the basic load profile. Note, that equal parameter settings only imply equal *levels* of switching probability and irrationality among customer groups, whereas the actual *decisions* made by each group still vary between groups.

[6] The p-value for equal profit share means of LinFA and Fixed at $q = 0.5$ in the left panel is $p = 0.0067$. In the right panel, $p = 0.6513$ ($\tau = 2.0$), $p = 0.5362$ ($\tau = 3.0$), and $p = 0.9690$ ($\tau = 6.0$). All other mean differences are statistically highly significant.

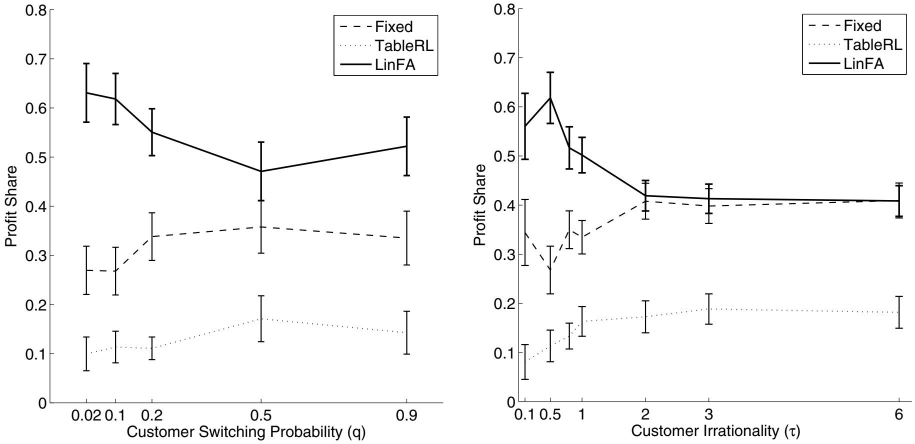


Fig. 3. Average profit share for LinFA, 70 runs per parameter combination, learning parameters $\alpha = 0.3$ (square root decay), $\epsilon = 0.3$ (linear decay), $\gamma = 1.0$, error bars indicate 95% confidence interval, $\tau = 0.5$ (left), $q = 0.1$ (right)

winning strategy: by raising tariff rates, a broker can increase its profit margin without affecting its customer base. The diminishing performance of LinFA here stems from an implicit assumption behind its manually constructed state space. Recall from Section 4.1 that LinFA's state space is constructed from the number of customers, the customer gradient, and its own profit margin. This is a well-chosen set of features for an environment where the broker should learn to attract additional customers conditional on positive margins. Yet, the random customer fluctuations in environments with large values of τ will be detrimental to such a broker's learning performance. In the next section, we will see how an alternative state space representation derived by AutoLinFA can be used to overcome this problem.

In further experiments we analyzed LinFA's performance for different simulation lengths, for different numbers of customers, for different values of the markup parameter μ of the reference strategies, for different settings of the learning parameters, and for different competitive settings including competition between multiple LinFA instances. We omit details here for the sake of brevity, but we do note that LinFA competes successfully in all cases except for pathological choices of learning parameters.

5.2 Offline Optimization

In our next experiment, we used a Genetic Algorithm to optimize AutoLinFA's state space and learning parameters for an environment with moderate customer switching probabilities ($q = 0.1$) and relatively rational customers ($\tau = 0.5$). We call the resulting broker instance AutoLinFA1. Interestingly, the optimization

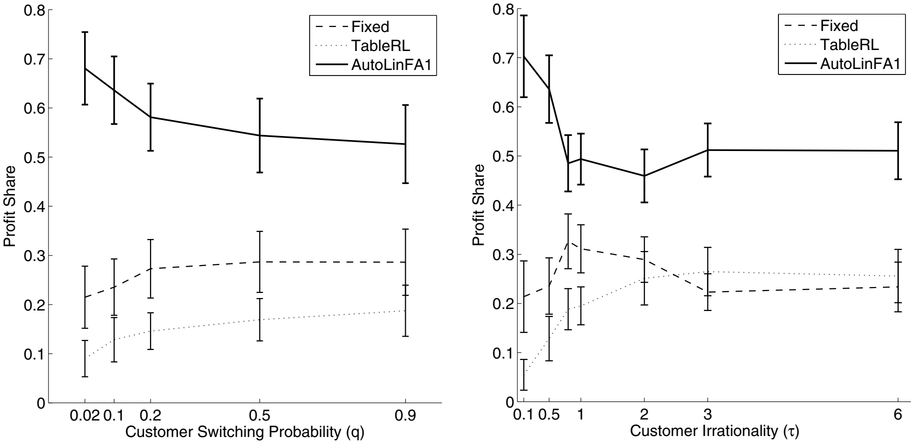


Fig. 4. Average profit share for AutoLinFA1, 70 runs per parameter combination, learning parameters $\alpha = 0.62$ (decaying at $t^{0.13}$), $\epsilon = 0.08$ (decaying at $t^{0.77}$), $\gamma = 0.85$, error bars indicate 95% confidence interval, $\tau = 0.5$ (left), $q = 0.1$ (right)

procedure chooses a very high-dimensional state space where 15 of the 29 candidate features are present, and a very high learning rate of $\alpha = 0.62$. This choice is a consequence of the comparatively stable environment for which AutoLinFA1 is optimized. A high level of environmental stability allows AutoLinFA1 to constantly adjust its policy to the current environment without running the risk of following pure chance developments. The result is not a single, overarching policy for different states of the environment, but a policy that *tracks*, and adjusts to, the current environmental state. This behavior is sometimes referred to as *non-associative* learning [13].

AutoLinFA1 performs better than LinFA, both for its target environment and for many other environments, as illustrated in Figure 4. It is important to note that these are *out-of-sample* results: we tested AutoLinFA1 in an environment with the same parameters, but different random influences on the wholesale market and on customer choice than in the offline training phase.

To confirm our findings, we optimized a second AutoLinFA instance, AutoLinFA2, for an environment where customers' tariff choices are much more irrational ($\tau = 2.0$). The corresponding performance results are given in Figure 5. AutoLinFA2's performance is again very strong for its target environment. These strong results come, however, at the cost of underperformance for market environments where customers act more rationally. In terms of learning parameters, the optimization procedure opted for a low learning rate of $\alpha = 0.03$, and higher exploration and discount rates ($\epsilon = 0.22$, $\gamma = 0.97$) as compared to the previous experiment. These choices are natural for an environment that is characterized by high degrees of uncertainty. The lower learning rate entails that actions must be rewarded many times before their likelihood of being selected rises in the learner's policy, and a large value of γ puts heavy emphasis on future

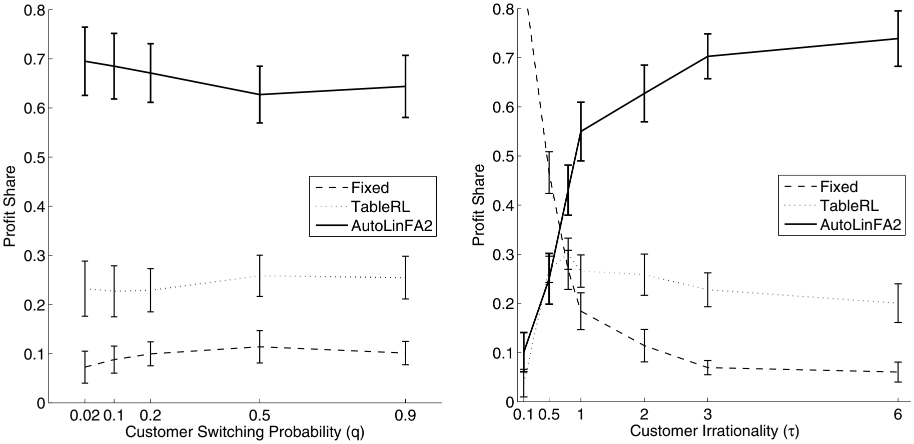


Fig. 5. Average profit share for AutoLinFA2, 70 runs per parameter combination, learning parameters $\alpha = 0.03$ (decaying at $t^{0.71}$), $\epsilon = 0.22$ (decaying at $t^{0.76}$), $\gamma = 0.97$, error bars indicate 95% confidence interval, $\tau = 2.0$ (left), $q = 0.5$ (right)

rewards as estimated by the inert action-value function. Together, these settings lead to a policy that is not easily swayed by random influences from the environment. The high exploration rate allows the broker to frequently deviate from its current optimal policy early in the simulation. This makes intuitive sense in an environment where exploration comes cheap (the high level of randomness lowers the value of acting greedily with respect to the current policy), and where it is potentially hard to find a good policy.

6 Related Work

To date, research on retail electricity trading has received relatively little attention. To our knowledge, Reddy et al. [12] were the first to suggest RL as an appropriate framework for constructing such brokers for retail electricity markets. A key distinguishing feature of the approach we present here is the automated, data-driven construction of the feature space. In contrast, the strategies developed in [12] are derived from manually constructed features and are limited in the number of economic signals they can accommodate as well as in their ability to incorporate new signals when the market environment changes. Another key distinction is that the brokers presented in [12] are derived for an environment with fixed rates of electricity consumption and production for all market participants where brokers source electricity exclusively from small-scale producers. Consequently, the broker agent learns to steer towards an optimal *consumer/producer ratio* among its subscribers by changing tariff rates. These settings yield a broker which is unable to develop appropriate responses to any variability of consumption and production over time or between different customers.

Reinforcement Learning has been used on a wide range of problems in electronic commerce in which agents aim to learn optimal policies through interaction with the environment. For example, [9] develop a data-driven approach for designing electronic auctions based on notions from RL. In the electricity domain, RL has primarily been used to derive wholesale trading strategies, or to build physical control systems. Examples of electricity wholesale applications include [5] and [10], who derive bidding strategies for electricity wholesale auctions. Physical control applications of RL include load and frequency control within the electric grid and autonomous monitoring applications, e.g. [15].

Whiteson et al. [18] provide interesting insights into the role of *environment overfitting* in empirical evaluations of Reinforcement Learning applications. They argue that *fitting*, i.e. the adaptation of a learner to environmental conditions known to be present in the target environment, is an appropriate strategy. *Overfitting*, i.e. the adaptation of the learner to conditions only present during evaluation, on the other hand, is inappropriate. These insights suggest that LinFA is a good general-purpose broker for settings in which little is known about customer characteristics in the target environment. Whenever prior knowledge is available, our offline optimization procedure is able to exploit this information and fit AutoLinFA brokers accordingly.

7 Conclusions

The Smart Grid vision relies critically on intelligent decentralized control mechanisms. In this paper, we explored a novel design for autonomous Electricity Broker Agents in future electricity retail markets.

We formalized a class of Smart Electricity Markets by means of a simulation model, and argued that our model represents the current state of the Smart Grid transition well. We then framed the broker problem as optimal control problem and used RL with function approximation to derive broker policies. We found that learning tariff-setting policies can be simplified significantly by normalizing tariff rates to the prevailing wholesale price, whereby strategies are formed with respect to profit margins. We demonstrated the efficacy of our broker design for a range of Smart Electricity Markets which varied substantially in terms of tariff choice behaviors among their customer populations. Our experimental results confirm that state space choice plays an important role in optimizing broker performance for a given environment, and that our brokers are significantly more flexible in this regard than previously suggested strategies.

In future work we aim to further explore the performance of our Electricity Broker Agent design in increasingly complex Smart Electricity Markets. Among the key features we aim to incorporate are advanced tariff structures, renewable energy sources, and customer models derived from behavioral economics. We believe that our proposed strategies can serve as an important benchmarks for future work and that this work offers a meaningful contribution to our understanding of key design decisions for broker agents to operate effectively in the Smart Grid.

References

1. Bichler, M., Gupta, A., Ketter, W.: Designing smart markets. Information Systems Research 21(4), 688–699 (2010)
2. Busoniu, L., Babuska, R., De Schutter, B., Ernst, D.: Reinforcement learning and dynamic programming using function approximators. CRC (2010)
3. ETPSG: European Technology Platform Smart Grids: Strategic deployment document for Europe's electricity networks of the future (April 2010)
4. European Commission: EU energy country factsheet (2011)
5. Gajjar, G., Khaparde, S., Nagaraju, P., Soman, S.: Application of actor-critic learning algorithm for optimal bidding problem of a genco. IEEE Transactions on Power Systems 18(1), 11–18 (2003)
6. Gottwalt, S., Ketter, W., Block, C., Collins, J., Weinhardt, C.: Demand side management - a simulation of household behavior under variable prices. Energy Policy 39, 8163–8174 (2011)
7. Herter, K., McAuliffe, P., Rosenfeld, A.: An exploratory analysis of california residential customer response to critical peak pricing of electricity. Energy 32(1), 25–34 (2007)
8. Ketter, W., Collins, J., Reddy, P., Flath, C., de Weerdt, M.: The power trading agent competition. Tech. Rep. ERS-2011-027-LIS, RSM Erasmus University, Rotterdam, The Netherlands (2011), http://ssrn.com/paper=1839139
9. Pardoe, D., Stone, P., Saar-Tsechansky, M., Keskin, T., Tomak, K.: Adaptive auction mechanism design and the incorporation of prior knowledge. INFORMS Journal on Computing 22(3), 353–370 (2010)
10. Rahimiyan, M., Mashhadi, H.: An adaptive q-learning algorithm developed for agent-based computational modeling of electricity market. IEEE Transactions on Systems, Man, and Cybernetics 40(5), 547–556 (2010)
11. Reddy, P., Veloso, M.: Learned behaviors of multiple autonomous agents in smart grid markets. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011 (2011)
12. Reddy, P., Veloso, M.: Strategy learning for autonomous agents in smart grid markets. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI, pp. 1446–1451 (2011)
13. Sutton, R., Barto, A.: Reinforcement learning: An introduction, vol. 116. Cambridge Univ. Press (1998)
14. Szepesvári, C.: Algorithms for reinforcement learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 4(1), 1–103 (2010)
15. Venayagamoorthy, G.: Potentials and promises of computational intelligence for smart grids. In: Power & Energy Society General Meeting, pp. 1–6. IEEE (2009)
16. de Weerdt, M., Ketter, W., Collins, J.: A theoretical analysis of pricing mechanisms and broker's decisions for real-time balancing in sustainable regional electricity markets. In: Conference on Information Systems and Technology, Charlotte, pp. 1–17 (November 2011)
17. Werbos, P.: Putting more brain-like intelligence into the electric power grid: What we need and how to do it. In: International Joint Conference on Neural Networks, pp. 3356–3359. IEEE (2009)
18. Whiteson, S., Tanner, B., Taylor, M.E., Stone, P.: Protecting against evaluation overfitting in empirical reinforcement learning. In: IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL) (April 2011)

Bayesian Nonparametric Inverse Reinforcement Learning

Bernard Michini and Jonathan P. How

Massachusetts Institute of Technology,
Cambridge, Massachusetts, USA
{bmich,jhow}@mit.edu

Abstract. Inverse reinforcement learning (IRL) is the task of learning the reward function of a Markov Decision Process (MDP) given the transition function and a set of observed demonstrations in the form of state-action pairs. Current IRL algorithms attempt to find a single reward function which explains the entire observation set. In practice, this leads to a computationally-costly search over a large (typically infinite) space of complex reward functions. This paper proposes the notion that if the observations can be partitioned into smaller groups, a class of much simpler reward functions can be used to explain each group. The proposed method uses a Bayesian nonparametric mixture model to automatically partition the data and find a set of simple reward functions corresponding to each partition. The simple rewards are interpreted intuitively as subgoals, which can be used to predict actions or analyze which states are important to the demonstrator. Experimental results are given for simple examples showing comparable performance to other IRL algorithms in nominal situations. Moreover, the proposed method handles cyclic tasks (where the agent begins and ends in the same state) that would break existing algorithms without modification. Finally, the new algorithm has a fundamentally different structure than previous methods, making it more computationally efficient in a real-world learning scenario where the state space is large but the demonstration set is small.

1 Introduction

Many situations in artificial intelligence (and everyday life) involve learning a task from observed demonstrations. In robotics and autonomy, there exists a large body of literature on the topic of learning from demonstration (see [1] for a survey). However, much of the robotics work has focused on generating direct functional mappings for low-level tasks. Alternatively, one might consider assuming a rational model for the demonstrator, and using the observed data to invert the model. This process can be loosely termed *inverse decision making*, and in practice it is often more challenging (both conceptually and computationally) than more direct mapping approaches. However, inverting the decision-making process may lend more insight as to the motivation of the demonstrator, and provide a richer explanation of the observed actions. Indeed, similar methodology has been increasingly used in psychology and cognitive science for action understanding and preference learning in humans [2, 3, 4, 5].

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 148–163, 2012.

If the problem is formally cast in the Markov decision process (MDP) framework, the rational model described above becomes an agent who attempts to maximize cumulative reward (in a potentially sub-optimal fashion). Inverse decision making becomes the problem of finding a state reward function that explains the observed state-action pairs of the agent, and is termed *inverse reinforcement learning* (IRL) in the seminal work of [6].

There have since been a variety of IRL algorithms developed [7, 8, 9, 10, 11, 12, 13]. These algorithms attempt to find one single reward function that explains the entirety of the observed demonstration set. This reward function must then be necessarily complex in order to explain the data sufficiently, especially when the task being demonstrated is itself complicated. Searching for a complex reward function is fundamentally difficult for two reasons. First, as the complexity of the reward model increases, so too does the number of free parameters needed to describe the model. Thus the search is over a larger space of candidate functions. Second, the process of testing candidate reward functions requires solving for the MDP value function (details in Section 2), the computational cost of which typically scales poorly with the size of the MDP state space, even for approximate solutions [14]. Thus finding a single, complex reward function to explain the observed demonstrations requires searching over a large space of possible solutions and substantial computational effort to test each candidate.

One potential solution to these problems would be to partition the observations into sets of smaller sub-demonstrations. Then, each sub-demonstration could be attributed to a smaller and less-complex class of reward functions. However, such a method would require manual partitioning of the data into an unknown number of groups, and inferring the reward function corresponding to each group.

The primary contribution of this paper is to present an IRL algorithm that automates this partitioning process using Bayesian nonparametric methods. Instead of finding a single, complex reward function, the demonstrations are partitioned and each partition is explained with a simple reward function. We assume a generative model in which these simple reward functions can be interpreted as *subgoals* of the demonstrator. The generative model utilizes a Chinese Restaurant Process (CRP) prior over partitions so that the number of partitions (and thus subgoals) need not be specified *a priori* and can be potentially infinite.

As discussed further in Section 5, a key advantage of this method is that the reward functions representing each subgoal can be extremely simple. For instance, one can assume that a subgoal is a single coordinate of the state space (or feature space). The reward function could then consist of a single positive reward at that coordinate, and zero elsewhere. This greatly constrains the space of possible reward functions, yet complex demonstrations can still be explained using a sequence of these simple subgoals. Also, the algorithm has no dependence on the sequential (i.e. temporal) properties of the demonstrations, instead focusing on partitioning the observed data by associated subgoal. Thus the resulting solution does not depend on the initial conditions of each demonstration, and moreover naturally handles cyclic tasks (where the agent begins and ends in the same state).

The paper proceeds as follows. Section 2 briefly covers preliminaries, and Section 3 describes the proposed algorithm. Section 4 presents experimental results comparing the proposed algorithm to existing IRL methods, and discussion is provided in Section 5.

2 Background

The following briefly reviews background material and notation necessary for the proposed algorithm. Throughout the paper, boldface is used to denote vectors subscripts are used to denote the elements of vectors (i.e. z_i is the ith element of vector $\boldsymbol{z}$).

2.1 Markov Decision Processes

A finite-state Markov Decision Process (MDP) is a tuple (S, A, T, γ, R) where S is a set of M *states*, A is a set of *actions*, $T : S \times A \times S \mapsto [0, 1]$ is the function of *transition probabilities* such that $T(s, a, s')$ is the probability of being in state s' after taking action a from state s, $R : S \mapsto \mathbb{R}$ is the *reward function*, and $\gamma \in [0, 1)$ is the *discount factor*.

A *stationary policy* is a function $\pi : S \mapsto A$. From [15] we have the following set of definitions and results:

1. The infinite-horizon expected reward for starting in state s and following policy π thereafter is given by the *value function* $V^\pi(s, R)$:

$$V^\pi(s, R) = E_\pi \left[\sum_{i=0}^{\infty} \gamma^i R(s_i) \middle| s_0 = s \right] \tag{1}$$

   The value function satisfies the following Bellman equation for all $s \in S$:

$$V^\pi(s, R) = R(s) + \gamma \left[\sum_{s'} T(s, \pi(s), s') V^\pi(s') \right] \tag{2}$$

   The so-called Q-function (or action-value function) $Q^\pi(s, a, R)$ is defined as the infinite-horizon expected reward for starting in state s, taking action a, and following policy π thereafter.

2. A policy π is optimal for M iff, for all $s \in S$:

$$\pi(s) = \operatorname*{argmax}_{a \in A} Q^\pi(s, a, R) \tag{3}$$

   An optimal policy is denoted as π^* with corresponding value function V^* and action-value function Q^*.

2.2 Inverse Reinforcement Learning

When inverse decision making is formally cast in the MDP framework, the problem is referred to as *inverse reinforcement learning* (IRL)[6]. An MDP/R is defined as a MDP for which everything is specified except the state reward function $R(s)$. Observations (demonstrations) are provided as a set of state-action pairs:

$$\boldsymbol{O} = \{(s_1, a_1), (s_2, a_2), ..., (s_N, a_N)\} \tag{4}$$

where each pair $O_i = (s_i, a_i)$ indicates that the demonstrator took action a_i while in state s_i. Inverse reinforcement learning algorithms attempt to find a reward function that rationalizes the observed demonstrations. For example, find a reward function $\widehat{R}(s)$ whose corresponding optimal policy π^* matches the observations $\boldsymbol{O}$.

It is clear that the IRL problem is ill-posed. Indeed, $\widehat{R}(s) = c \ \ \forall s \in S$, where c is any constant, will make any set of state-action pairs O trivially optimal. Also, O may contain inconsistent or conflicting state-action pairs, i.e. (s_i, a_1) and (s_i, a_2) where $a_1 \neq a_2$. Furthermore, the "rationality" of the demonstrator is not well-defined (e.g., is the demonstrator perfectly optimal, and if not, to what extent sub-optimal).

Most existing IRL algorithms attempt to resolve the ill-posedness by making some assumptions about the form of the demonstrator's reward function. For example, in [7] it is assumed that the reward is a sum of weighted state features, and finds a reward function to match the demonstrator's feature expectations. In [8] a linear-in-features reward is also assumed, and a maximum margin optimization is used to find a reward function that minimizes a loss function between observed and predicted actions. In [9] it is posited that the demonstrator samples from a prior distribution over possible reward functions, and thus Bayesian inference is used to find a posterior over rewards given the observed data. An implicit assumption in these algorithms is that the demonstrator is using a single, fixed reward function.

The three IRL methods mentioned above (and other existing methods such as [10, 11, 13]) share a generic algorithmic form, which is given by Algorithm 1, where the various algorithms use differing definitions of "similar" in Step 2c. We note that each iteration of the algorithm requires re-solving for the optimal MDP value function in Step 2a, and the required number of iterations (and thus MDP solutions) is potentially unbounded.

2.3 Chinese Restaurant Process Mixtures

Since the proposed IRL algorithm seeks to partition the observed data, a Chinese restaurant process (CRP) is used to define a probability distribution over the space of possible partitions. The CRP proceeds as follows:

1. The first customer sits at the first table.
2. Customer i arrives and chooses the first unoccupied table with probability $\frac{\eta}{i-1+\eta}$, and an occupied table with probability $\frac{c}{i-1+\eta}$, where c is the number of customers already sitting at that table.

Algorithm 1. Generic inverse reinforcement learning algorithm

```
GenericIRL(MDP/$R$, Observations $O_{1:N}$, Reward representation $\widehat{R}(s|w)$)
 1. Initialize reward function parameters $w^0$
 2. Iterate from $t = 1$ to $T$:
    (a) Solve for optimal MDP value function $V^*$ corresponding to reward function
        $\widehat{R}(s|w^{(t-1)})$
    (b) Use $V^*$ to define a policy $\widehat{\pi}$.
    (c) Choose parameters $w^{(T)}$ to make $\widehat{\pi}$ more similar to demonstrations $O_{1:N}$ in
        the next iteration.
 3. Return Reward function given by $\widehat{R}(s|w^{(T)})$
```

The concentration hyperparameter η controls the probability that a customer starts a new table. Using $z_i = j$ to denote that customer i has chosen table j, C_j to denote the number of customers sitting at table j, and J_{i-1} to denote the number of tables currently occupied by the first $i-1$ customers, the assignment probability can be formally defined by:

$$P(z_i = j|z_{1\ldots i-1}) = \begin{cases} \frac{C_J}{i-1+\eta} & j \leq J_{i-1} \\ \frac{\eta}{i-1+\eta} & j = J_{i-1}+1 \end{cases} \tag{5}$$

This process induces a distribution over table partitions that is *exchangeable* [16], meaning that the order in which the customers arrive can be permuted and any partition with the same proportions will have the same probability. A Chinese restaurant process mixture is defined using the same construct, but each table is endowed with parameters θ of a probability distribution which generates data points x_i:

1. Each table j is endowed with parameter θ_j, where θ_j is drawn i.i.d. from a prior $P(\theta)$.
2. For each customer i that arrives:
   (a) The customer sits at table j according to (5) (the assignment variable $z_i = j$).
   (b) A datapoint x_i is drawn i.i.d. from $P(x|\theta_j)$.

Thus each datapoint x_i has an associated table assignment $z_i = j$ and is drawn from the distribution $P(x|\theta_j)$. Throughout the paper we use i to index state-action pairs O_i of the demonstrator ("customers" in the CRP analogy). We use j to index partitions of the state-actions pairs ("tables" in the CRP analogy). Finally, the table parameters θ_j in the CRP mixture model presented above correspond to the simple reward function for each partition, which we interpret as *subgoals* throughout the paper.

3 Bayesian Nonparametric IRL Algorithm

The following section describes the Bayesian nonparametric subgoal IRL algorithm. We start with two definitions necessary to the algorithm.

Definition 1. *A state subgoal g is simply a single coordinate $g \in S$ of the MDP state space. The associated state subgoal reward function $R_g(s)$ is:*

$$R_g(s) = \begin{cases} c \text{ at state } g \\ 0 \text{ at all other states} \end{cases} \tag{6}$$

where c is a positive constant.

While the notion of a state subgoal and its associated reward function may seem trivial, a more general *feature* subgoal will be defined in the following sections to extend the algorithm to a feature representation of the state space.

Definition 2. *An MDP agent in state s_i moving towards some state subgoal g chooses an action a_i with the following probability:*

$$P(a_i|s_i, g) = \pi(a_i|s_i, g) = \frac{e^{\alpha Q^*(s_i, a_i, R_g)}}{\sum_a e^{\alpha Q^*(s_i, a, R_g)}} \tag{7}$$

Thus π defines a stochastic policy as in [15], and is essentially our model of rationality for the demonstrating agent (this is the same rationality model as in [9] and [4]). In Bayesian terms, it defines the likelihood of observed action a_i when the agent is in state s_i. The hyperparameter α represents our degree of confidence in the demonstrator's ability to maximize reward.

3.1 Generative Model

The set of observed state-action pairs O defined by (4) are assumed to be generated by the following model. The model is based on the likelihood function above, but adds a CRP partitioning component. This addition reflects our basic assumption that the demonstrations can be explained by partitioning the data and finding a simple reward function for each partition.

An agent finds himself in state s_i (because of the Markov property, the agent need not consider how he got to s_i in order to decide which action a_i to take). In analogy to the CRP mixture described in Section 2.3, the agent chooses which partition a_i should be added to, where each existing partition j has its own associated subgoal g_j. The agent can also choose to assign a_i to a new partition whose subgoal will be drawn from the base distribution $P(g)$ of possible subgoals. The assignment variable z_i is set to denote that the agent has chosen partition z_i, and thus subgoal g_{z_i}. As in equation (5), $P(z_i|z_{1:i-1}) = CRP(\eta, z_{1:i-1})$. Now that a partition (and thus subgoal) has been selected for a_i, the agent generates the action according to the stochastic policy $a_i \sim \pi(a_i|s_i, g_{z_i})$ from equation (7).

The joint probability over $O_{1:N}$, $\boldsymbol{z}$, and $\boldsymbol{g}$ is given below, since it will be needed to derive the conditional distributions necessary for sampling:

$$\begin{aligned} P(O_{1:N}, \boldsymbol{z}, \boldsymbol{g}) &= P(O_{1:N}|\boldsymbol{z}, \boldsymbol{g})\ P(\boldsymbol{z}, \boldsymbol{g}) &(8)\\ &= P(O_{1:N}|\boldsymbol{z}, \boldsymbol{g})\ P(\boldsymbol{z})\ P(\boldsymbol{g}) &(9)\\ &= \prod_{i=1}^{N} \underbrace{P(O_i|g_{z_i})}_{\text{likelihood}} \underbrace{P(z_i|z_{-i})}_{\text{CRP}} \prod_{j=1}^{J_N} \underbrace{P(g_j)}_{\text{prior}} &(10) \end{aligned}$$

where (9) follows since subgoal parameters g_j for each new partition are drawn independently from prior $P(g)$ as described above. As shown in (10), there are three key elements to the joint probability. The likelihood term is the probability of taking each action a_i from state s_i given the associated subgoal g_{z_i}, and is defined in (7). The CRP term is the probability of each partition assignment z_i given by (5). The prior term is the probability of each partition's subgoal (J_N is used to indicate the number of partitions after observing N datapoints). The subgoals are drawn i.i.d. from discrete base distribution $P(g)$ each time a new partition is started, and thus have non-zero probability given by $P(g_j)$.

The model assumes that O_i is conditionally independent of O_j for $i \neq j$ given g_{z_i}. Also, it can be verified that the CRP partition probabilities $P(z_i|z_{-i})$ are exchangeable. Thus, the model implies that the data O_i are exchangeable [16]. Note that this is weaker than implying that the data are independent and identically distributed (i.i.d.). The generative model instead assumes that there is an underlying grouping structure that can be exploited in order to decouple the data and make posterior inference feasible.

The CRP partitioning allows for an unknown and potentially infinite number of subgoals. By construction, the CRP has "built-in" complexity control, i.e. its concentration hyperparameter η can be used to make a smaller number of partitions more likely.

3.2 Inference

The generative model (10) has two sets of hidden parameters, namely the partition assignments z_i for each observation O_i, and the subgoals g_j for each partition j. Thus the job of the IRL algorithm will be to infer the posterior over these hidden variables, $P(\boldsymbol{z}, \boldsymbol{g}|O_{1:N})$. While both $\boldsymbol{z}$ and $\boldsymbol{g}$ are discrete, the support of $P(\boldsymbol{z}, \boldsymbol{g}|O_{1:N})$ is combinatorially large (since $\boldsymbol{z}$ ranges over the set of all possible partitions of N integers), so exact inference of the posterior is not feasible. Instead, approximate inference techniques must be used. Gibbs sampling [17] is in the family of Markov chain Monte Carlo (MCMC) sampling algorithms and is commonly used for approximate inference of Bayesian nonparametric mixture models [18, 19, 20]. Since we are interested in the posterior of both the assignments and subgoals, uncollapsed Gibbs sampling is used where both the $\boldsymbol{z}$ and $\boldsymbol{g}$ are sampled in each sweep.

Each Gibbs iteration involves sampling from the conditional distributions of each hidden variable given all of the other variables (i.e. sample one unknown at a time with all of the others fixed). Thus the conditionals for each partition assignment z_i and subgoal g_j must be derived.

The conditional for partition assignment z_i can be derived as follows:

$$\begin{aligned} P(z_i|z_{-i}, \boldsymbol{g}, \boldsymbol{O}) &\propto P(z_i, O_i \mid z_{-i}, O_{-i}) && (11)\\ &= P(z_i|z_{-i}, \boldsymbol{g}, O_{-i})P(O_i|z_i, z_{-i}, \boldsymbol{g}, O_{-i}) && (12)\\ &= P(z_i|z_{-i}) \;\; P(O_i|z_i, z_{-i}, \boldsymbol{g}, O_{-i}) && (13)\\ &= \underbrace{P(z_i|z_{-i})}_{\text{CRP}} \;\; \underbrace{P(O_i|g_{z_i})}_{\text{likelihood}} && (14) \end{aligned}$$

where (11) is the definition of conditional probability, (12) applies the chain rule, (13) follows from the fact that assignment z_i depends only on the other assignments z_{-i}, and (14) follows from the fact that each O_i depends only on its assigned subgoal g_{z_i}.

Algorithm 2. Bayesian nonparametric IRL

```
BNIRL(MDP/$R$, Observations $O_{1:N}$, Confidence $\alpha$, Concentration $\eta$)
 1. for each unique $s_i \in O_{1:N}$:
    (a) Solve for and store $V^*(R_g)$, where $g = s_i$ and $R_g$ is defined by (20)
    (b) Sample an initial subgoal $g_1^{(0)}$ from prior $P(g)$ and set all assignments $z_i^{(0)} = 1$
 2. for $t = 1$ to $T$:
    (a) for each current subgoal $g_j^{(t-1)}$: Sample subgoal $g_j^{(t)}$ from (17)
    (b) for each observation $O_i \in O_{1:N}$: Sample assignment $z_i^{(t)}$ from (14)
 3. Return samples $z^{(1:T)}$ and $g^{(1:T)}$, discarding samples for burn-in and lag if desired
```

When sampling from (14), the exchangeability of the data is utilized to treat z_i as if it was the last point to be added. Probabilities (14) are calculated with z_i being assigned to each existing partition, and for the case when z_i starts a new partition with subgoal drawn from the prior $P(g)$. While the number of partitions is potentially infinite, there will always be a finite number of groups when the length of the data N is finite, so this sampling step is always feasible.

The conditional for each partition's subgoal g_j is derived as follows:

$$\begin{aligned} P(g_j|\boldsymbol{z}, \boldsymbol{O}) &\propto P(O_{I_j}|g_j, \boldsymbol{z}, O_{-I_j})P(g_j|\boldsymbol{z}, O_{-I_j}) && (15)\\ &= \sum_{i \in I_j} P(O_i|g_{z_i})\ P(g_j|\boldsymbol{z}, O_{-I_j}) && (16)\\ &= \sum_{i \in I_j} \underbrace{P(O_i|g_{z_i})}_{\text{likelihood}}\ \underbrace{P(g_j)}_{\text{prior}} && (17)\end{aligned}$$

where (15) applies Bayes' rule, (16) follows from the fact that each O_i depends only on its assigned subgoal g_{z_i}, and (17) follows from the fact that the subgoal g_j of each partition is drawn i.i.d. from the prior over subgoals. The index set $I_j = \{i : z_i = j\}$.

Sampling from (17) depends on the form of the prior over subgoals $P(g)$. When the subgoals are assumed to take the form of *state subgoals* (Definition 1), then $P(g)$ is a discrete distribution whose support is the set S of all states of the MDP. In this paper, we propose the following simplifying assumption to increase the efficiency of the sampling process.

Proposition 1. *The prior $P(g)$ is assumed to have support only on the set S_O of MDP states, where $S_O = \{s \in S : s = s_i$ for some observation $O_i = (s_i, a_i)\}$.*

This proposition assumes that the set of all possible subgoals is limited to only those states encountered by the demonstrator. Intuitively it implies that during the demonstration, the demonstrator achieves each of his subgoals. This is not the same as assuming a perfect demonstrator (the expert is not assumed to get to each subgoal optimally, just eventually). Sampling of (17) now scales with the number of unique states in the observation set $O_{1:N}$. While this proposition may seem limiting, the experimental results in Section 4 indicate that it does not affect performance compared to other IRL algorithms and greatly reduces the required amount of computation. Algorithm 2 defines the proposed Bayesian nonparametric inverse reinforcement learning method. The algorithm outputs samples which form a Markov chain whose stationary distribution is the posterior, so that sampled assignments $\boldsymbol{z}^{(T)}$ and subgoals $\boldsymbol{g}^{(T)}$ converge to a sample from the true posterior $P(\boldsymbol{z}, \boldsymbol{g}|O_{1:N})$ as $T \to \infty$ [17, 21]. Note that instead of solving for the MDP value function in each iteration (as is typical with IRL algorithms, see Algorithm 1), Algorithm 2 pre-computes all of the necessary value functions. The number of required value functions is upper bounded by the number of elements in the support of the prior $P(g)$. When we assume Proposition 1, then the support of $P(g)$ is limited to the set of unique states in the observations $O_{1:N}$. Thus the required number of MDP solutions scales with the size of the observed data set $O_{1:N}$, *not* with the number of required iterations. We see this as an advantage in a learning scenario when the size of the MDP is potentially large but the amount of demonstrated data is small.

3.3 Convergence in Expected 0-1 Loss

To demonstrate convergence, it is common in IRL to define a loss function which in some way measures the difference between the demonstrator and the predictive output of the algorithm [8, 9, 10]. In Bayesian nonparametric IRL, the assignments $\boldsymbol{z}$ and subgoals $\boldsymbol{g}$ represent the hidden variables of the demonstration that must be learned. Since these variables are discrete, a 0-1 loss function is suitable:

$$\mathcal{L}\left[(\boldsymbol{z}, \boldsymbol{g}), (\widehat{\boldsymbol{z}}, \widehat{\boldsymbol{g}})\right] = \begin{cases} 1 \text{ if } \ (\widehat{\boldsymbol{z}}, \widehat{\boldsymbol{g}}) = (\boldsymbol{z}, \boldsymbol{g}) \\ 0 \text{ otherwise} \end{cases} \tag{18}$$

The loss function evaluates to 1 if the estimated parameters $(\widehat{\boldsymbol{z}}, \widehat{\boldsymbol{g}})$ are exactly equal to the true parameters $(\boldsymbol{z}, \boldsymbol{g})$, and 0 otherwise. We would like to show that, for the Bayesian nonparametric IRL algorithm (Algorithm 2), the expected value of the loss function (18) given a set of observations $O_{1:N}$ is minimized as the number of iterations T increases. Theorem 1 establishes this.

Theorem 1. *Assuming observations $O_{1:N}$ are generated according to the generative model defined by (10), the expected 0–1 loss defined by (18) is minimized by the empirical mode of the samples $(\boldsymbol{z}^{(1:T)}, \boldsymbol{g}^{(1:T)})$ output by Algorithm 2 as the number of iterations $T \to \infty$.*

Proof. It can be verified that the maximum *a posteriori* (MAP) parameter values, defined here by

$$(\widehat{\boldsymbol{z}}, \widehat{\boldsymbol{g}}) = \underset{(\boldsymbol{z},\boldsymbol{g})}{\operatorname{argmax}}\, P(\boldsymbol{z}, \boldsymbol{g}|O_{1:N})$$

minimize the expected 0–1 loss defined in (18) given the observations $O_{1:N}$ (see [22]). By construction, Algorithm 2 defines a Gibbs sampler whose samples $(\boldsymbol{z}^{(1:T)}, \boldsymbol{g}^{(1:T)})$ converge to samples from the true posterior $P(\boldsymbol{z}, \boldsymbol{g}|O_{1:N})$ so long as the Markov chain producing the samples is ergodic [17]. From [23], a sufficient condition for ergodicity of the Markov chain in Gibbs sampling requires only that the conditional probabilities used to generate samples are non-zero. For Algorithm 2, these conditionals are defined by (14) and (17). Since clearly the likelihood (7) and CRP prior (5) are always non-zero, then the conditional (14) is always non-zero. Furthermore, the prior over subgoals $P(g)$ is non-zero for all possible g by assumption, so that (17) is non-zero as well.

Thus the Markov chain is ergodic and the samples $(\boldsymbol{z}^{(1:T)}, \boldsymbol{g}^{(1:T)})$ converge to samples from the true posterior $P(\boldsymbol{z}, \boldsymbol{g}|O_{1:N})$ as $T \to \infty$. By the strong law of large numbers, the empirical mode of the samples, defined by

$$(\widetilde{\boldsymbol{z}}, \widetilde{\boldsymbol{g}}) = \underset{(\boldsymbol{z}^{(1:T)},\boldsymbol{g}^{(1:T)})}{\operatorname{argmax}}\, P(\boldsymbol{z}, \boldsymbol{g}|O_{1:N})$$

converges to the true mode $(\widehat{\boldsymbol{z}}, \widehat{\boldsymbol{g}})$ as $T \to \infty$, and this is exactly the MAP estimate of the parameters which was shown to minimize the 0–1 loss. □

We note that, given the nature of the CRP prior, the posterior will be multimodal (switching partition indices does not affect the partition probability even though the numerical assignments $\boldsymbol{z}$ will be different). As such, the argmax above is used to define the *set* of parameter values which maximize the posterior. In practice, the sampler need only converge on one of these modes to find a satisfactory solution.

The rate at which the loss function decreases relies on the rate the empirical sample mode(s) converges to the true mode(s) of the posterior. This is a property of the approximate inference algorithm and, as such, is beyond the scope of this paper (convergence properties of the Gibbs sampler have been studied, for instance in [24]). As will be seen empirically in Section 4, the number of iterations required for convergence is typically similar to (or less than) that required for other IRL methods.

3.4 Action Prediction

IRL algorithms find reward models with the eventual goal of learning to predict what action the agent will take from a given state. As in Algorithm 1, the typical output of the IRL algorithm is a single reward function that can be used to define a policy which predicts what action the demonstrator would take from a given state.

In Bayesian nonparametric IRL (Algorithm 2), in order to predict action a_k from state s_k, a subgoal must first be chosen from the mode of the samples $\widehat{\boldsymbol{g}} = \text{mode}(\boldsymbol{g}^{(1:T)})$. This is done by finding the most likely partition assignment z_k after marginalizing over actions using Equation (11):

$$z_k = \underset{z_i}{\text{argmax}} \sum_a P\left(z_i \mid \widehat{z}_{-i}, \widehat{\boldsymbol{g}}, O_k = (s_k, a)\ \right) \tag{19}$$

where $\widehat{\boldsymbol{z}}$ is the mode of samples $\boldsymbol{z}^{(1:T)}$. Then, an action is selected using the policy defined by (7) with $\widehat{g}_{z_k}$ as the subgoal.

Alternatively, the subgoals can simply be used as waypoints which are followed in the same order as observed in the demonstrations. In addition to predicting actions, the subgoals in $\widehat{\boldsymbol{g}}$ can be used to analyze which states in the demonstrations are important, and which are just transitory.

3.5 Extension to Discrete Feature Spaces

Linear combinations of state features are commonly used in reinforcement learning to approximately represent the value function in a lower-dimensional space [14, 15]. Formally, a k-dimensional feature vector is a mapping $\Phi : S \mapsto \mathbb{R}^k$. Likewise, a discrete k-dimensional feature vector is a mapping $\Phi : S \mapsto \mathbb{Z}^k$, where $\mathbb{Z}$ is the set of integers.

Many of the IRL algorithms listed in Section 2.2 assume that the reward function can be represented as a linear combination of features. We extend Algorithm 2 to accommodate discrete feature vectors by defining a *feature subgoal* in analogy to the state subgoal from Definition 1.

Definition 3. *Given a k-dimensional discrete feature vector Φ, a feature subgoal $g(\boldsymbol{f})$ is the set of states in S which map to the coordinate $\boldsymbol{f}$ in the feature space. Formally,*

$g(\boldsymbol{f}) = \{s \in S : \Phi(s) = \boldsymbol{f}\}$ where $\boldsymbol{f} \in \mathbb{Z}^k$. The associated feature subgoal reward function $R_{g(\boldsymbol{f})}(s)$ is defined as follows:

$$R_{g(f)}(s) = \begin{cases} c, & s \in g(\boldsymbol{f}) \\ 0, & s \notin g(\boldsymbol{f}) \end{cases} \tag{20}$$

where c is a positive constant.

From this definition it can be seen that a state subgoal is simply a specific instance of a feature subgoal, where the features are binary indicators for each state in S. Algorithm 2 runs exactly as before, with the only difference being that the support of the prior over reward functions $P(g)$ is now defined as the set of unique feature coordinates induced by mapping S through ϕ. Proposition 1 is also still valid should we wish to again limit the set of possible subgoals to only those feature coordinates in the observed demonstrations, $\Phi(s_{1:N})$. Finally, feature subgoals do not modify any of the assumptions of Theorem 1, thus convergence is still attained in 0-1 loss.

4 Experiments

Experimental results are given for three test cases. All three use a 20 × 20 Grid World MDP (total of 400 states) with walls. Note that while this is a relatively simple MDP, it is similar in size and nature to experiments done in the seminal papers of each of the compared algorithms. Also, the intent of the experiments is to compare basic properties of the algorithms in nominal situations (as opposed to finding the limits of each).

The agent can move in all eight directions or choose to stay. Transitions are noisy, with probability 0.7 of moving in the chosen direction. The discount factor $\gamma = 0.99$, and value iteration is used to find the optimal value function for all of the IRL algorithms tested. The demonstrator in each case makes optimal decisions based on the true reward function. While this is not required for Bayesian nonparametric IRL, it is an assumption of one of the other algorithms tested [7]. In all cases, the 0-1 policy loss function is used to measure performance. The 0-1 policy loss simply counts the number of times that the learned policy (i.e. the optimal actions given the learned reward function) does not match the demonstrator over the set of observed state-action pairs.

4.1 Grid World

The first example uses the state-subgoal Bayesian nonparametric IRL algorithm. The prior over subgoal locations is chosen to be uniform over states visited by the demonstrator (as in Proposition 1). The demonstrator chooses optimal actions towards each of three subgoals $(x, y) = \{(10, 12), (2, 17), (2, 2)\}$, where the next subgoal is chosen only after arrival at the current one. Figure 3 shows the state-action pairs of the demonstrator (left), the 0-1 policy loss averaged over 25 runs (center), and the posterior mode of subgoals and partition assignments (colored arrows denote assignments to the corresponding colored boxed subgoals) after 100 iterations (right). The algorithm reaches a minimum in loss after roughly 40 iterations, and the mode of the posterior subgoal locations converges to the correct coordinates. We note that while the subgoal locations have correctly converged after 100 iterations, the partition assignments for each state-action pair have not yet converged for actions whose subgoal is somewhat ambiguous.

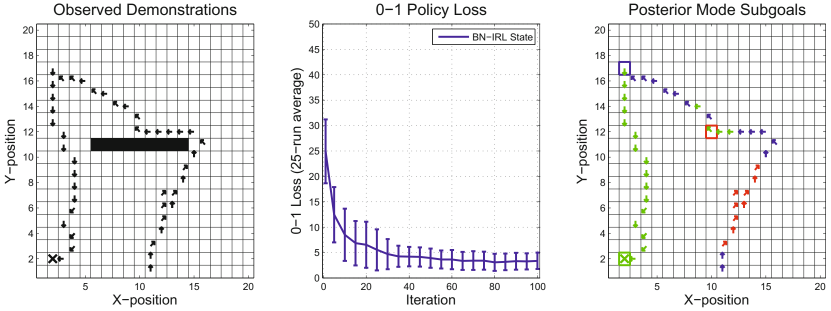


Fig. 3. Observed state-action pairs for simple grid world example (left), where arrows indicate direction of the chosen action and X's indicate choosing the "stay" action. 0-1 policy loss for Bayesian nonparametric IRL (center). Posterior mode of subgoals and partition assignments (right). Colored arrows denote assignments to the corresponding colored boxed subgoals.

4.2 Grid World with Features Comparison

In the next test case, Bayesian nonparametric IRL (for both state- and feature-subgoals) is compared to three other IRL algorithms, using the same Grid World setup as in Section 4.1: "Abbeel" IRL using the quadratic program variant [7], Maximum Margin Planning using a loss function that is non-zero at states not visited by the demonstrator [8], and Bayesian IRL [9]. A set of six features $\phi_{1:6}(s)$ are used, where feature k has an associated state s_{ϕ_k}. The value of feature k at state s is simply the Manhattan distance (1-norm) from s to s_{ϕ_k}:

$$\phi_k(s) = ||s - s_{\phi_k}||_1 \tag{21}$$

The true reward function is defined as $R(s) = \boldsymbol{w}^T \boldsymbol{\phi}(s)$ where $\boldsymbol{w}$ is a vector of randomly-chosen weights. The observations consist of five demonstrations starting at state $(x, y) = (15, 1)$, each having 15 actions which follow the optimal policy corresponding to the true reward function. Note that this dataset satisfies the assumptions of the three compared algorithms, though it does not strictly follow the generative process of Bayesian nonparametric IRL. Figure 4 shows the state-action pairs of the demonstrator (left) and the 0-1 policy loss, averaged over 25 runs versus iteration for each algorithm (right). All but Bayesian IRL achieve convergence to the same minimum in policy loss by 20 iterations, and Bayesian IRL converges at roughly 100 iterations (not shown). Even though the assumptions of the Bayesian nonparametric IRL were not strictly satisfied (the assumed model (10) did not generate the data), both the state- and feature-subgoal variants of the algorithm achieved performance comparable to the other IRL methods.

Table 1 compares average initialization and per-iteration run-times for each of the algorithms. These are given only to show general trends, as the Matlab implementations of the algorithms were by no means optimized for efficiency. The initialization of Bayesian nonparametric IRL takes much longer than the others, since during this period the algorithm is pre-computing optimal value functions for each of the possible subgoal locations (i.e. each of the states encountered by the demonstrator). However, the Bayesian nonparametric IRL per-iteration time is roughly an order of magnitude less than the other algorithms, since the others must re-compute an optimal value function each iteration.

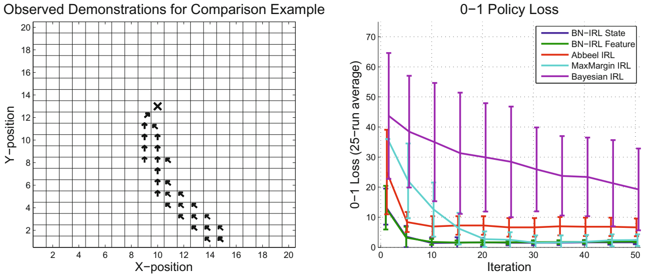


Fig. 4. Observed state-action pairs for grid world comparison example (left). Comparison of 0-1 Policy loss for various IRL algorithms (right).

Table 1. Run-time comparison for various IRL algorithms

	Initialization (sec)	Per-iteration (sec)
BN-IRL	15.3	0.21
Abbeel-IRL	0.42	1.65
MaxMargin-IRL	0.41	1.16
Bayesian-IRL	0.56	3.27

4.3 Grid World with Loop

In the final experiment, five demonstrations are generated using subgoals as in Section 4.1. The demonstrator starts in state $(x, y) = (10, 1)$, and proceeds to subgoals $(x, y) = \{(19, 9), (10, 17), (2, 9), (10, 1)\}$. Distance features (as in Section 4.2) are placed at each of the four subgoal locations. Figure 5 (left) shows the observed state-action pairs. This dataset clearly violates the assumptions of all three of the compared algorithms, since more than one reward function is used to generate the state-action pairs. However, the assumptions are violated in a reasonable way. The data resemble a common robotics scenario in which an agent leaves an initial state, performs some tasks, and then returns to the same initial state.

Figure 5 (center) shows that the three compared algorithms, as expected, do not converge in policy loss. Both Bayesian nonparametric algorithms, however, perform almost exactly as before and the mode posterior subgoal locations converge to the four true subgoals (Figure 5 right). Again, the three compared algorithms would have worked properly if the data had been generated by a single reward function, but such a reward function would have to be significantly more complex (i.e. by including temporal elements). Bayesian nonparametric IRL is able to explain the demonstrations without modification or added complexity.

5 Discussion

5.1 Comparison to Existing Algorithms

The example in Section 4.2 shows that, for a simple problem, Bayesian nonparametric IRL performs comparably to existing algorithms in cases where the data are generated using a single reward function. Approximate computational run-times indicate that

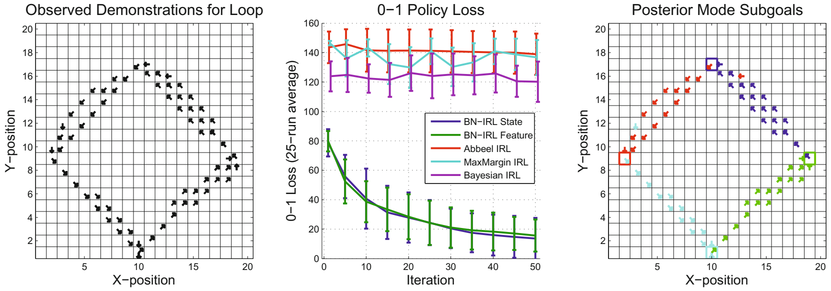


Fig. 5. Observed state-action pairs for grid world loop example (left). Comparison of 0-1 Policy loss for various IRL algorithms (center). Posterior mode of subgoals and partition assignments (right).

overall required computation is similar to existing algorithms. As noted in Section 3.2, however, Bayesian nonparametric IRL solves for the MDP value function once for each unique state in the demonstrations. The other algorithms solve for the MDP value function once per iteration. We see this fundamental difference as an advantage which will make the new algorithm scalable to real-world domains where the size of the state space is large and the set of demonstrations is small. Testing in these larger domains is an area of ongoing work.

The loop example in Section 4.3 highlights several fundamental differences between Bayesian nonparametric IRL and existing algorithms. While the example resembles the fairly-common traveling salesman problem, it breaks the fundamental assumption of existing IRL methods that the demonstrator is optimizing a *single* reward function. These algorithms could be made to properly handle the loop case, but not without added complexity or manual partitioning of the demonstrations. Bayesian nonparametric IRL, on the other hand, is able to explain the loop example without any modifications. The ability of the new algorithm to automatically partition the data and explain each group with a simple subgoal reward eliminates the need to find a single, complex temporal reward function. Furthermore, the Chinese restaurant process prior naturally limits the number of partitions in the resulting solution, rendering a parsimonious explanation of the data.

5.2 Related and Future Work

Aside from the relation to existing IRL methods, we see a connection to option methods for MDPs [25]. While the original work explains how to *use* options to perform potentially complex tasks in an MDP framework, Bayesian nonparametric IRL could be used to *learn* options from demonstrations. Options in this case would take the form of optimal policies corresponding to each of the learned subgoal rewards. Exploration of the connection to option learning is an avenue of future work.

There are several other areas of ongoing and future work. First, the results given here are for simple problems and are by no means exhaustive. Ongoing work seeks to apply the method in more complex robotics domains where the size of the state space is significantly larger, and the observations are generated by an actual human demonstrator. Also, Bayesian nonparametric IRL could be applied to higher-level planning problems where the list of subgoals found by the algorithm may be useful in more richly analyzing the human demonstrator.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems 57(5), 469–483 (2009)
2. Kautz, H., Allen, J.F.: Generalized plan recognition. In: Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 32–37. AAAI (1986)
3. Verma, D., Rao, R.: Goal-Based Imitation as Probabilistic Inference over Graphical Models. In: Advances in Neural Information Processing Systems 18, vol. 18, pp. 1393–1400 (2006)
4. Baker, C.L., Saxe, R., Tenenbaum, J.B.: Action understanding as inverse planning.. Cognition 113(3), 329–349 (2009)
5. Jern, A., Lucas, C.G., Kemp, C.: Evaluating the inverse decision-making approach to preference learning. Processing, 1–9 (2011)
6. Ng, A.Y., Russell, S.: Algorithms for inverse reinforcement learning. In: Proc. of the 17th International Conference on Machine Learning, pp. 663–670 (2000)
7. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Twentyfirst International Conference on Machine learning ICML 2004, p. 1 (2004)
8. Ratliff, N.D., Bagnell, J.A., Zinkevich, M.A.: Maximum margin planning. In: Proc. of the 23rd International Conference on Machine Learning, pp. 729–736 (2006)
9. Ramachandran, D., Amir, E.: Bayesian inverse reinforcement learning. In: IJCAI, pp. 2586–2591 (2007)
10. Neu, G., Szepesvari, C.: Apprenticeship learning using inverse reinforcement learning and gradient methods. In: Proc. UAI (2007)
11. Syed, U., Schapire, R.E.: A Game-Theoretic Approach to Apprenticeship Learning. In: Advances in Neural Information Processing Systems 20, vol. 20, pp. 1–8 (2008)
12. Ziebart, B.D., Maas, A., Bagnell, J.A., Dey, A.K.: Maximum Entropy Inverse Reinforcement Learning. In: Proc. AAAI, pp. 1433–1438. AAAI Press (2008)
13. Lopes, M., Melo, F., Montesano, L.: Active Learning for Reward Estimation in Inverse Reinforcement Learning. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 31–46. Springer, Heidelberg (2009)
14. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. The Optimization and Neural Computation Series, vol. 5. Athena Scientific (1996)
15. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
16. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis. Texts in statistical science, vol. 2. Chapman & Hall/CRC (2004)
17. Geman, S., Geman, D.: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6(6), 721–741 (1984)
18. Sudderth, E.B.: Graphical Models for Visual Object Recognition and Tracking by. Thesis 126(1), 301 (2006)
19. Escobar, M.D., West, M.: Bayesian density estimation using mixtures. Journal of the American Statistical Association 90(430), 577 (1995)
20. Neal, R.M.: Markov Chain Sampling Methods for Dirichlet Process Mixture Models. Journal Of Computational And Graphical Statistics 9(2), 249 (2000)
21. Andrieu, C., De Freitas, N., Doucet, A., Jordan, M.I.: An Introduction to MCMC for Machine Learning. Science 50(1), 5–43 (2003)

22. Berger, J.O.: Statistical Decision Theory and Bayesian Analysis. Springer Series in Statistics. Springer (1985)
23. Neal, R.M.: Probabilistic Inference Using Markov Chain Monte Carlo Methods. Intelligence 62, 144 (1993)
24. Roberts, G.O., Sahu, S.K.: Updating schemes, correlation structure, blocking and parameterisation for the Gibbs sampler. Journal of the Royal Statistical Society - Series B: Statistical Methodology 59(2), 291–317 (1997)
25. Sutton, R.S., Precup, D., Singh, S.: Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence 112(1-2), 181–211 (1999)

Bootstrapping Monte Carlo Tree Search with an Imperfect Heuristic

Truong-Huy Dinh Nguyen, Wee-Sun Lee, and Tze-Yun Leong

National University of Singapore, Singapore 117417
{trhuy,leews,leongty}@comp.nus.edu.sg

Abstract. We consider the problem of using a heuristic policy to improve the value approximation by the Upper Confidence Bound applied in Trees (UCT) algorithm in non-adversarial settings such as planning with large-state space Markov Decision Processes. Current improvements to UCT focus on either changing the action selection formula at the internal nodes or the rollout policy at the leaf nodes of the search tree. In this work, we propose to add an auxiliary arm to each of the internal nodes, and always use the heuristic policy to roll out simulations at the auxiliary arms. The method aims to get fast convergence to optimal values at states where the heuristic policy is optimal, while retaining similar approximation as the original UCT at other states. We show that bootstrapping with the proposed method in the new algorithm, UCT-Aux, performs better compared to the original UCT algorithm and its variants in two benchmark experiment settings. We also examine conditions under which UCT-Aux works well.

1 Introduction

Monte Carlo Tree Search (MCTS) [1], or more specifically Upper Confidence Bound applied in Trees (UCT) [2], is a state-of-the-art approach to solving large state-space planning problems. Example applications of the UCT algorithm in the games domain include Go [1,3,4], General Game Playing [5], Real-Time Strategy Game [6], etc.

The algorithm estimates the value of a state by building a search tree using simulated episodes, or *rollouts*, via interactions with the simulator. Instead of sampling every branch equally, the goal is to focus samplings in tree branches that are more promising. In particular, UCT achieves that by choosing the action, or *arm*, with the highest estimated upper bound to simulate at every internal node, and randomly selects actions after leaving the tree to finish the rollout.

Because UCT uses random sampling to discover nodes with good return, it could take a long time to achieve good performance. To address this problem, many enhancements have been used to improve the search control of the algorithm by either (1) tweaking the action selection formula at the internal nodes [7,3,5,4,8], and/or (2) designing better-informed rollout policies in place of random sampling at the leaf nodes [3,4].

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 164–179, 2012.

We consider the problem of using a heuristic function to improve the approximated value function computed by UCT. Taking the approaches above, the first method is to initialize new tree nodes with heuristic values and the second is to use the chosen heuristic to roll out simulations at the leaf nodes. As intended, these two methods could greatly influence the search control by guiding it into more promising regions that are determined by the heuristic. However, when the heuristic function does not accurately reflect the prospect of the states, it could feed the algorithm with false information, thereby leading the search into regions that should be kept unexplored otherwise.

In this work, we propose a novel yet simple enhancement method. Given a heuristic in the form of an imperfect policy π, the method adds an additional arm at every internal node of the search tree. This special arm is labeled by the action suggested by π and once selected, rolls out the rest of the sampling episode using π. If the policy π works well at a state, we expect it to quickly give a good estimate of the value of the state without relying too much on the other arms. The method aims to get fast convergence to optimal values at states where the heuristic policy is optimal, while retaining similar approximation as the original UCT at other states.

We compared this method with two aforementioned techniques in two domains, namely Obstructed Sailing, an extension of the original Sailing problem previously used to measure UCT's performance in [2], and Sheep Savior, a large state-space MDP that characterizes a generic two-player collaborative puzzle game. The results showed that UCT-Aux the new algorithm (**Aux** for *auxiliary* arms) significantly outperforms its competitors when coupled with reasonable heuristics.

One nice property of this method is that it does not affect the implementation of other bootstrapping techniques: No modification of the action selection formula nor the rollout policy at any leaf nodes except for the added arms is required. This allows different sources of bootstrapping knowledge to be combined into one algorithm for more performance boost.

The rest of the paper is structured as follows. We first give a brief overview of MDP, UCT and its popular enhancements before presenting UCT-Aux. Next, we describe two experimental setups for comparing the agents' performance and analyze the results. We also identify the common properties of the heuristics used in two experimental domains and provide some insights on why UCT-Aux works well in those cases. Finally, we conclude the paper by discussing the possible usage of UCT-Aux.

2 Background

2.1 Markov Decision Process

A Markov Decision Process characterizes a planning problem with tuple (S, A, T, R, γ), in which

- S is the set of all states,

- A is the set of all available actions,
- $T_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action $a \in A$ in state $s \in S$ at time t will lead to state $s' \in S$ at time $t + 1$,
- $R_a(s, s')$ is the immediate reward received after the state transition from s to s' triggered by action a, and
- $\gamma \in [0, 1]$ is the discount factor; in infinite-horizon settings, γ must be strictly smaller than 1.

An action *policy* π is a function, possibly stochastic, that returns an action $\pi(s)$ for every state $s \in S$. In infinite-horizon discounted MDPs, the objective is to choose an action policy π^* that maximizes some cumulative function of the received rewards, typically the expected discounted sum $\sum_{t=0}^{\infty} \gamma^t R_{a^*}(s_t, s_{t+1})$. An MDP can be effectively solved using different methods, one of which is the value iteration algorithm based on the Bellman's equation [9]. The algorithm maintains a value function $V(s)$, where s is a state, and iteratively updates the value function using the equation

$$V_{t+1}(s) = \max_a \left(\sum_{s'} T_a(s, s')(R_a(s, s') + \gamma V_t(s')) \right).$$

This value iteration algorithm is guaranteed to converge to the optimal value function $V^*(s)$, which gives the optimal expected cumulative reward of running the optimal policy from state s.

The optimal value function V^* can be used to construct the optimal policy by taking action a^* in state s such that $a^* = \operatorname{argmax}_a \left\{ \sum_{s'} T_a(s, s') V^*(s') \right\}$. The optimal Q-function is constructed from V^* as follows:

$$Q^*(s, a) = \sum_{s'} T_a(s, s')(R_a(s, s') + \gamma V^*(s')).$$

$Q^*(s, a)$ denotes the maximum expected long-term reward of an action a when executed in state s.

One key issue that hinders MDPs and Value Iteration from being widely used in real-life planning tasks is the large state space size (usually exponential in the number of state variables) that is often required to model realistic problems.

2.2 Upper Confidence Bound Applied to Trees (UCT)

UCT [2] is an anytime algorithm that approximates the state-action value in real time using Monte Carlo simulations. It was inspired by Sparse Sampling [10], the first near-optimal policy whose runtime does not depend on the size of the state space. The approach is particularly suitable for solving planning problems with very large or possibly infinite state spaces.

The algorithm searches forward from a given starting state, building up a tree whose nodes alternate between reachable future states and state-action pairs (Figure 1). State nodes are called *internal* if their child state-action pairs have been expanded and *leaf* otherwise. Starting with a root state, the algorithm

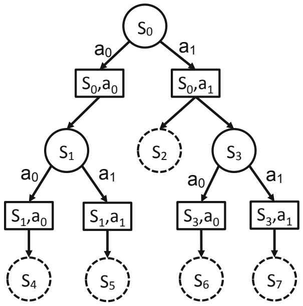


Fig. 1. A sample UCT search tree with two valid actions a_0 and a_1 at any state. Circles are *state* nodes and rectangles are *state-action* nodes; solid state nodes are *internal* while dotted are *leafs*.

iteratively rolls out simulations from this root node; each time an internal node is encountered, it is regarded as a multi-armed bandit and UCB1 [11] is used to determine the action or *arm* to sample, i.e., the edge to traverse. In particular, at an internal node s, the algorithm selects an action according to

$$\pi_{UCT}(s) = \underset{a}{\operatorname{argmax}} \left\{ Q_{UCT}(s,a) + 2C_p \sqrt{\frac{\log n(s)}{n(s,a)}} \right\}, \tag{1}$$

in which

- $Q_{UCT}(s,a)$ is the estimated value of state-action pair (s,a), taken to be the weighted average of its children's values.
- $C_p > 0$ is a suitable hand-picked constant.
- $n(s)$ is the total number of rollouts starting from s.
- $n(s,a)$ is the number of rollouts that execute a at s.

At the chosen child state-action node, the simulator is randomly sampled for a next state with accompanying reward; new states automatically become leaf nodes. From the leaf nodes, rollouts are continued using random sampling until a termination condition is satisfied, such as reaching terminal states or simulation length limit. Once finished, the returned reward propagates up the tree, with the value at each parent node being the weighted average of its child nodes' values; suppose the rollout executes action a at state s and accumulates reward $R(s,a)$ in the end.

- at state-action nodes, $n(s,i) = n(s,i) + 1$ and $Q_{UCT}(s,a) = Q_{UCT}(s,a) + \frac{1}{n(s,a)}(R(s,a) - Q_{UCT}(s,a))$
- at state nodes, $n(s) = n(s) + 1$.

Typically one leaf node is converted to internal per rollout, upon which its child state-action nodes are generated. When the algorithm is terminated, the root's arm with highest $Q_{UCT}(s,a)$ is returned[1].

[1] In practice, returning the arm with highest $n(s,a)$ is also a common choice.

When used for infinite-horizon discounted MDPs, the search can be cut off at an ϵ_0-horizon. Given any $\epsilon > 0$, with ϵ_0 small enough, the algorithm is proven to converge to the arm whose value is within the ϵ-vicinity of the optimal arm [2].

2.3 Enhancement Methods

In vanilla UCT, new state-action nodes are initialized with uninformed default values and random sampling is used to finish the rollout when leaving the tree. Given a source of prior knowledge, Gelly and Silver [3] proposed two directions to bootstrap UCT:

1. Initialize new action-state nodes with $n(s,a) = n_{prior}(s,a)$ and $Q_{UCT}(s,a) = Q_{prior}(s,a)$, and
2. Replace random sampling by better-informed exploration guided by π_{prior}.

We refer to these two algorithms as UCT-I (UCT with new nodes **i**nitialized to heuristic values) and UCT-S (UCT with **s**imulations guided by π_{prior}); UCT-IS is the combination of both methods. UCT-I and UCT-S can be further tuned using domain knowledge to mitigate the flaw of a bad heuristic and amplify the influence of a good one by adjusting the dependence of the search control on the heuristic at internal nodes. In this work, we do not investigate the effect of such tuning to ensure a fair comparison between techniques when employed as is.

In the same publication [3], the authors proposed another bootstrapping technique, namely Rapid Action Value Estimation (RAVE), which we do not examine in this work. The technique is specifically designed for domains in which an action from a state s has similar effect regardless of when it is executed, either at s or after many moves. RAVE uses the All-Moves-As-First (AMAF) heuristic [12] instead of $Q_{UCT}(s,a)$ in Equation 1 to select actions. Many board games such as Go or Breakthrough [5] have this desired property. In our experiment domains, RAVE is not applicable, because the actions are mostly directional movements, e.g., $\{N, E, S, W\}$, thus tied closely to the state they are performed at.

3 UCT-Aux: Algorithm

Given an added policy π, we propose a new algorithm UCT-Aux that follows the same search control as UCT except for two differences.

1. At every internal node s, besides $|A(s)|$ normal arms with $A(s)$ being the set of valid actions at state s, an additional arm labeled by the action $\pi(s)$ is created (Figure 2).
2. When this arm is selected by Equation 1, it stops expanding the branch but rolls out a simulation using π; value update is carried out from the auxiliary arm up to the root as per normal.

The method aims to better manage mixed-quality heuristics. If the heuristic π's value estimation at a state is good, we expect the added arm to dominate the

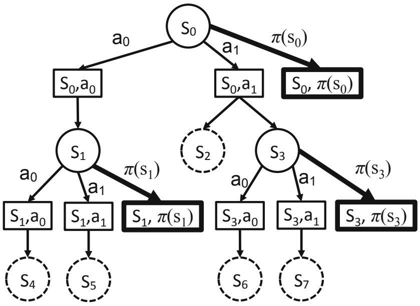


Fig. 2. Sample search tree of UCT-Aux

distribution of rollouts and quickly give a good estimate of the state's value without the need to inspect other arms. Otherwise, the search control will focus rollouts in ordinary arms, thus retaining similar approximation as vanilla UCT.

For stochastic heuristic policies, at every internal node, not one but κ auxiliary arms are added, with κ being the number of actions a_π such that $P(\pi(s) = a_\pi) > 0$. As such, the number of arms at internal nodes is bounded by $2|A|$ since $\kappa \leq |A|$.

3.1 Convergence Analysis

We will show that regardless of the added policy's quality, UCT-Aux converges in finite-horizon MDPs[2]. The proof follows closely that of UCT analysis by treating the auxiliary arms as any other ordinary arms. As a recap, UCT convergence analysis revolves around the analysis of non-stationary multi-armed bandits with reward sequences satisfying some drift conditions, which is proven to be the case for UCT's internal nodes with appropriate choice of bias sequence C_p[3]. In particular, the drift conditions imposed on the payoff sequences go as follows:

- The expected values of the averages $\overline{X}_{in} = \frac{1}{n}\sum_{t=1}^{n} X_{it}$ must converge for all arms i with n being the number of pulls and X_{it} the payoff of pull t. Let $\mu_{in} = E[\overline{X}_{in}]$ and $\mu_i = lim_{n\to\infty}\mu_{in}$.
- $C_p > 0$ can be chosen such that the tail inequalities $P(\overline{X}_{i,n(i)} \geq \mu_i + c_{t,n(i)}) \leq t^{-4}$ and $P(\overline{X}_{i,n(i)} \leq \mu_i - c_{t,n(i)}) \leq t^{-4}$ are satisfied for $c_{t,n(i)} = 2C_p\sqrt{\frac{\ln t}{n(i)}}$ with $n(i)$ being the number of times arm i is pulled up to time t.

Firstly, we will show that all internal nodes of UCT-Aux have arms yielding rewards satisfying the drift conditions. Suppose the horizon of the MDP is D, the number of actions per state is K and the heuristic policy is deterministic ($\kappa = 1$); this can be proven using induction on D. Note that i.i.d. payoff sequences satisfy the drift conditions trivially due to Hoeffding's inequality.

[2] As mentioned in [2], for use with discounted infinite-horizon MDPs, the search tree can be cut off at the effective ϵ_0-horizon with ϵ_0 being the desired accuracy at root.

[3] Empirically, C_p is often chosen to be an upper bound of the accumulated reward starting from the current state.

- $D = 1$: Suppose the root has already been expanded, i.e., become internal. It has $K + 1$ arms, which either lead to leaf nodes (ordinary arms) or return i.i.d. payoffs (auxiliary arm). Since leaf nodes have i.i.d. payoffs, all arms satisfy drift conditions.
- $D > 1$: Assume that all internal state nodes under the root have arms satisfying the drift conditions, e.g., s_1 and s_3 in Figure 2. Consider any ordinary arm of the root node (the added arm's payoff sequence is already i.i.d.), for instance, (s_0, a_1). Its payoff average is the weighted sum of payoff sequences in all leafs and state-action nodes on the next two levels of the subtree, i.e., leaf s_2, arms (s_3, a_0), (s_3, a_1) and $(s_3, \pi(s_3))$, all of which satisfy drift conditions due to either the inductive hypothesis or producing i.i.d. payoffs. Theorem 4 in [2] posits that the weighted sum of payoff sequences conforming to drift conditions also satisfies drift conditions; therefore, all arms originating from the root node satisfy drift conditions.

As a result, the theorems on non-stationary bandits in [2] hold for UCT-Aux's internal nodes as well. Therefore, we can obtain similar results to Theorem 6 of [2], with the difference being statistical measures related to the auxiliary arms such as μ_{aux} and Δ_{aux}, i.e., the new algorithm's probability of selecting a suboptimal arm converges to zero as the number of rollouts tends to infinity.

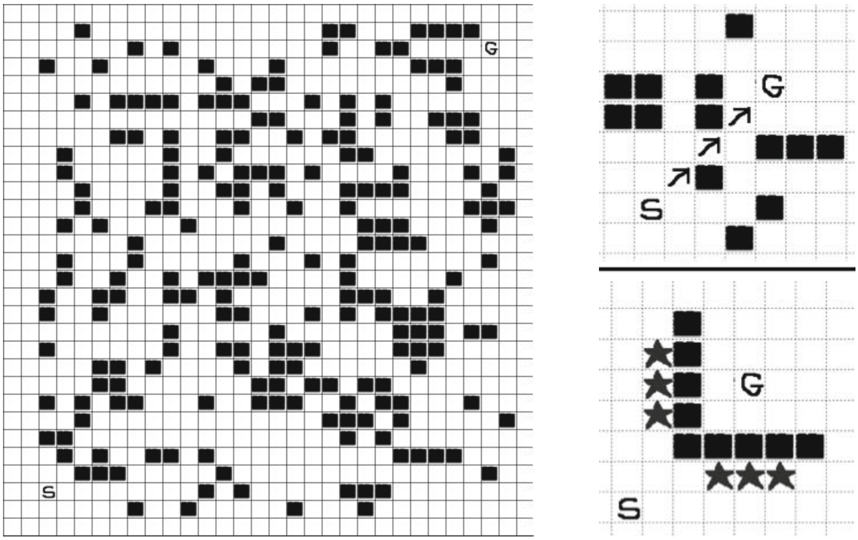
4 Experiments

We compare the performance of UCT-Aux against UCT, UCT-I, UCT-S and UCT-IS in two domains: Obstructed Sailing and Sheep Savior. Obstructed Sailing extends the benchmark Sailing domain by placing random blockage in the map; the task is to quickly move a boat from one point to a destination on a map, disturbed by changing wind, while avoiding obstacles. Sheep Savior features a two-player maze game in which the players need to herd a sheep into its pen while protecting it from being killed by two ghosts in the same environment.

5 Obstructed Sailing

The Sailing domain, originally used to evaluate the performance of UCT [2], features a control problem in which the planner is tasked to move a boat from a starting point to a destination under certain disturbing wind conditions. In our version, there are several obstacles placed randomly in the map (see Figure 3a).

In this domain, the state is characterized by tuple $\langle x, y, b, w_{prev}, w_{curr} \rangle$ with (x, y) being the current boat position, b the current boat posture or direction, w_{prev} the previous wind direction and w_{curr} the current wind direction. Directions take values in $\{N, NE, E, SE, S, SW, W, NW\}$, i.e. clockwise starting from North. The controller's valid action set includes all but the directions against w_{curr}, out of the map or into an obstacle. After each time step, the wind has roughly equal probability to remain unchanged, switch to its left or its right [13].



(a) Obstructed Sailing sample map

(b) SailingTowardsGoal

Fig. 3. Obstructed Sailing domain; (a) a randomized starting configuration, (b) SailingTowardsGoal heuristic produces near-optimal estimates/policies in good cases but misleads the search control in others

Depending on the relative angle between the action taken and w_{curr}, a cost from 1 to 4 minutes is incurred. Additionally, changing from a port to a starboard tack or vice versa causes a tack delay of 3 minutes. In total, an action can cost anywhere from 1 to 7 minutes, i.e., $C_{min} = 1$ and $C_{max} = 7$ [13]. We model the problem as an infinite-horizon discounted MDP with discount factor $\gamma = 0.99$.

5.1 Choice of Heuristic Policies

A simple heuristic for this domain is to select a valid action that is closest to the direction towards goal position regardless of the cost, thereafter referred to as SailingTowardsGoal. For instance, in the top subfigure of Figure 3b, at the starting state marked by "S", if current wind is not SW, SailingTowardsGoal will move the boat in the NE direction; otherwise, it will execute either N or E.

This heuristic is used in UCT-I and UCT-IS by initializing new state-action nodes with values

$$n_{STG}(s,a) \leftarrow 1$$

$$Q_{STG}(s,a) \leftarrow C(s,a) + C_{min}\frac{1-\gamma^{d(s',g)+1}}{1-\gamma}$$

with $C(s,a)$ being the cost of executing action a at state s and $d(s',g)$ the minimum distance between next state s' and goal position g. The initialized value can be seen as the minimum cost incurred when all future wind directions are favorable for desired movement. For UCT-S, the random rollouts are replaced by $\pi(s) = \operatorname{argmax}_a Q_{STG}(s,a)$.

Heuristic Quality. This heuristic works particularly well for empty spaces, producing near-optimal plans if there are no obstacles. However, it could be counterproductive when encountering obstacles. In the bottom subfigure of Figure 3b, if a rollout from the starting position is guided by SailingTowardsGoal, it could be stuck oscillating among the starred tiles, thus giving inaccurate estimation of the optimal cost.

5.2 Setup and Results

The trial map size is 30 by 30, with fixed starting and goal positions at respectively $(2, 2)$ and $(27, 27)$ (Figure 3a). We generated 100 randomized instances of the map, where obstacles are shuffled by giving each grid tile $p = 0.4$ chance to be blocked[4]. Each instance is tried five times, each of which with different starting boat postures and wind directions.

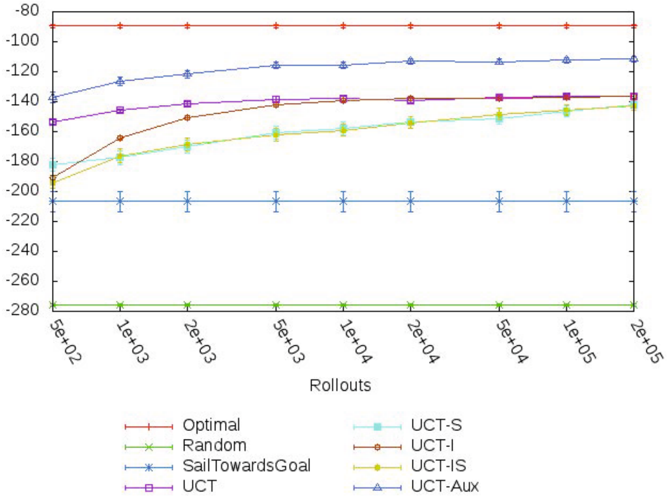


Fig. 4. Performance comparison of UCT, UCT-S, UCT-I, UCT-IS and UCT-Aux when coupled with the heuristic SailingTowardsGoal; y-axis is the reward average with error bars being the standard errors of the means

All UCT variants (UCT, UCT-I, UCT-S, UCT-IS and UCT-Aux) use the same $C_p = C_{max}/(1-\gamma) = 700$ and the search horizon[5] is set to be 300; an optimal path should not be very far from 60 steps as most actions move the boat

[4] We tried with different values of $p \in \{0.05, 0.1, 0.2, 0.3, 0.5\}$ and they all yield similar results as Figure 4; the detailed charts are not presented due to space constraint.

[5] The search horizon is chosen to be long enough so that the cost accumulated after the horizon has small effect to the total cost.

closer to the goal. The exact optimal policy is obtained using Value Iteration. Note that the performance of Optimal agent varies because of the randomization of starting states (initial boat and wind direction) and map configurations.

Given the same number of samplings, UCT-Aux outperforms all competing UCT variants, despite the mixed quality of the added policy SailingTowardsGoal when dealing with obstacles (Figure 4). Note that without parameter tuning, both UCT-I and UCT-S are inferior to vanilla UCT, but between UCT-I and UCT-S, UCT-I shows faster performance improvement when the number of samplings increases. The reason is because when SailingTowardsGoal produces inaccurate heuristic values, UCT-I only suffers at early stage while UCT-S endures misleading guidance until the search reaches states where the policy yields more accurate heuristic values. The heuristic's impact is stronger in UCT-S than UCT-I: UCT-IS's behavior is closer to UCT-S than UCT-I.

6 Sheep Savior

This domain is an extension of the *Collaborative Ghostbuster* game introduced in [14] as the testbed for their assistance framework for collaborative games. The game features two players (a shepherd and a dog) whose task is to herd a sheep into its pen while avoiding it to be killed by two ghosts in a maze-like environment. All non-player characters (NPCs) run away from the players within a certain distance, otherwise the ghosts chase the sheep and the sheep runs away from ghosts. Since ghosts can only be shot by the Shepherd, the dog's role is strictly to gather the NPCs (Figure 5).

Both protagonists have 5 movement actions (no_move, N, S, E and W) while Shepherd has an additional action to inflict damage on a nearby ghost, hence a total of 30 compound actions. The two players are given rewards for successfully killing ghosts (5 points) or herding sheep into its pen (10 points). If the sheep is killed, the game is terminated with penalty -10. The discount factor in this domain is set to be 0.99.

6.1 Choice of Heuristic Policies

The game can be seen as having three subtasks, each of which is the task of catching a single ghost or herding a single sheep, as shown in Figure 5. Each of these subtasks consists of only two players and one NPC, hence has manageable complexity and can be solved exactly offline using Value Iteration.

A heuristic Q-value can be obtained by taking the average of all individual subworlds', or subtasks', Q-values, as an estimate for one state-action pair's value. Specifically, at state s the policy, GoalAveraging, yields

$$n_{GA}(s, a) \leftarrow 1$$

$$Q_{GA}(s, a) = \frac{1}{m} \sum_{i=1}^{m} Q_i(s_i, a)$$

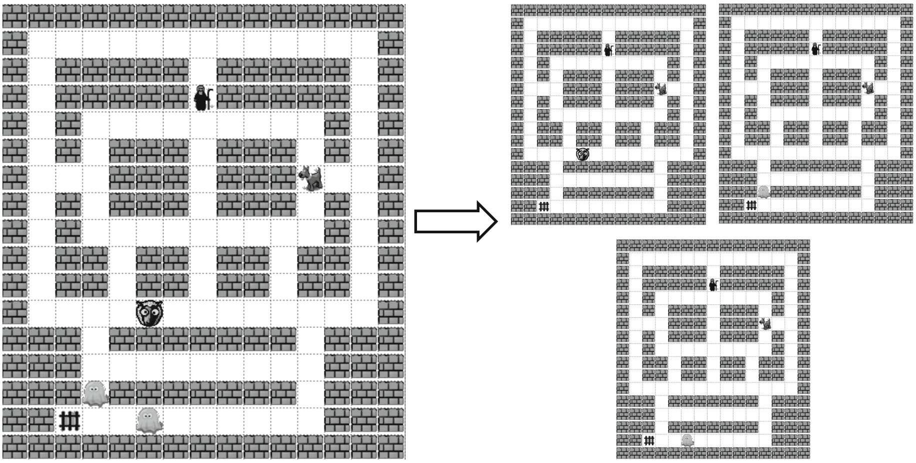


Fig. 5. Task decomposition in Sheep Savior

in which s_i is the projection of s in subtask i, m is the number of subtasks, i.e. three in this case, and $Q_i(s_i, a)$ are subtasks' Q-values. The corresponding heuristic policy can be constructed as $\pi_{GA}(s) = \text{argmax}_a\, Q_{GA}(s, a)$.

Heuristic Quality. GoalAveraging works well in cases when the sheep is well-separated from ghosts. However, when these creatures are close to each other, the policy's action estimation is no longer valid and could yield deadly results. The under-performance is due to the fact that the heuristic is oblivious to the interactivity between subtasks, in this case, ghost-killing-sheep scenarios.

6.2 Setup and Results

The map shown in Figure 5 is tried 200 times, each of which with a different randomized starting configurations. We compare the means of discounted rewards produced by the following agents: Random, GoalAveraging, UCT, UCT-I, UCT-S, and UCT-Aux. The optimal policy in this domain is not computed due to the prohibitively large state space, i.e., $104^5 * 3^2 \approx 10^{11}$ since each ghost has at most two health points. All UCT variants have a fixed planning depth of 300. In our setup, one second of planning yields roughly 200 rollouts on average, so we do not run simulations with higher numbers of rollouts than 10000 due to time constraint. Moreover, in this game-related domain, the region of interest is in the vicinity of 200 to 500 rollouts for practical use.

As shown in Figure 6, UCT-Aux outperforms the other variants, especially early on with small numbers of rollouts. UCT-S takes advantage of GA better than UCT-I, which yields even worse performance than vanilla UCT. Observing the improvement rate of UCT-S we expect it to approach UCT-Aux much sooner than others, although asymptotically all of them will converge to the same optimal value when enough samplings are given and the search tree is sufficiently expanded; the time taken could be prohibitively long though.

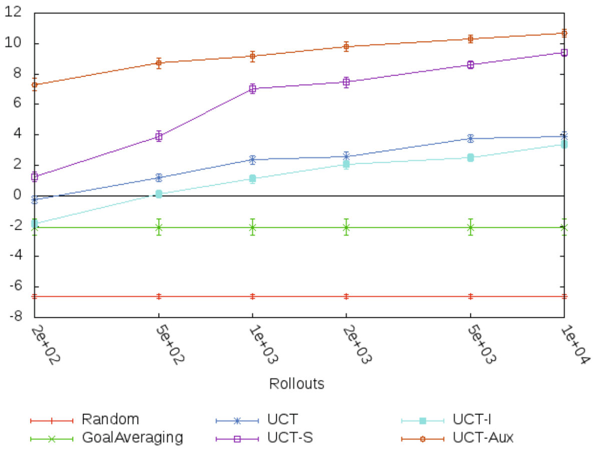


Fig. 6. Performance comparison of Random, GoalAveraging, UCT, UCT-S, UCT-I, and UCT-Aux when coupled with Goal Averaging

7 Discussions

Although UCT-Aux shows superior performances in the experiments above, we observe that the chosen heuristic policies share a common property that is crucial for UCT-Aux's success: they show behaviors of "*make it or break it*". In other words, at most states s, their action value estimate $Q_\pi(s,a)$ is either near-optimal or as low as that of a random policy $Q_{rand}(s,a)$.

Specifically, in Obstructed Sailing, if following SailingTowardsGoal can bring the boat from a starting state to goal position, e.g., when the line of sight connecting source and destination points lies entirely in free space, the resultant course of actions does not deviate much from the optimal action sequences. However when the policy fails to reach the goal, it could be stuck fruitlessly. For instance, Figure 3b depicts one such case; once the boat has reached either one of three starred tiles underneath the goal position, unless at least three to five wind directions in a row are E, SailingTowardsGoal results in oscillating the boat among these starred tiles. The resultant cost is therefore very far from optimal and could be as low as the cost incurred by random movement.

In contrast, an example for heuristics that are milder in nature is the policy StochasticOptimal.0.2 which issues optimal actions with probability 0.2 and random actions for the rest. This policy is also suboptimal but almost always yields better estimation than random movement; it is not as "*extreme*" as SailingTowardsGoal. Figure 7, which charts the performance histograms of StochasticOptimal.0.2 alongside with SailingTowardsGoal, shows that a majority of runs with SailingTowardsGoal yield costs that are either optimal or worse than Random's.

Similarly, GoalAveraging in Sheep Savior is also extreme: By ignoring the danger of Ghosts when around Sheep, it is able to quickly herd the Sheep in the Pen or kill nearby Ghosts (good), or end the game prematurely by forcing the Sheep into

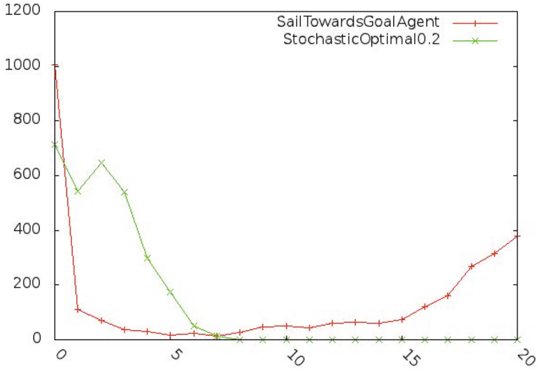


Fig. 7. Performance histograms of heuristics in Obstructed Sailing. The returned costs of a heuristic are allocated relatively into bins that equally divide the cost difference between Random and Optimal agents; x-axis denotes the bin number and y-axis the frequency.

Ghosts' zones (bad). We hypothesize that one way to obtain extreme heuristics is by taking near-optimal policies of the relaxed version of the original planning problem, in which aspects of the environment that cause negative effects to the accumulated reward are removed. For instance, SailingTowardsGoal is in spirit the same as the optimal policy for maps with no obstacle, while GoalAveraging should work well if the ghosts do not attack sheep.

As UCT-Aux is coupled with heuristic policies with this "extreme" characteristic, rollouts are centralized at auxiliary arms of states where $\pi(s)$ is near-optimal, and distributed to ordinary arms otherwise. Consequently, the value estimation falls back to the default random sampling where π produces inaccurate estimates instead of relying entirely on π as does UCT-S.

7.1 When Does UCT-Aux Not Work?

Figure 8 charts the worst-case behavior of UCT-Aux when the coupled heuristic's estimate is mostly better than random sampling but much worse than that of the optimal policy, e.g. the heuristic StochasticOptimal.0.2 in Obstructed Sailing.

The reason behind UCT-Aux's flop is the same as that of UCT, i.e., due to the overly "optimism" of UCB1, as described in [8]. At each internal node, samplings are directed into suboptimal arms that appear to perform the best so far, exponentially more than the rest (Theorem 1 [2]) when convergence has not started. Even though each arm is guaranteed to be sampled an infinite number of times when the number of samplings goes to infinity (Theorem 3 [2]), the sub-polynomial rate means only a tiny fraction of samplings are spent on attempting bad-looking arms. As a result, in a specially designed binary tree search case, UCT takes at least $\Omega(exp(exp(...exp(2)...)))$ samplings before the optimal node is discovered; the term is a composition of $D-1$ exponential functions with D being the number of actions in the optimal sequence.

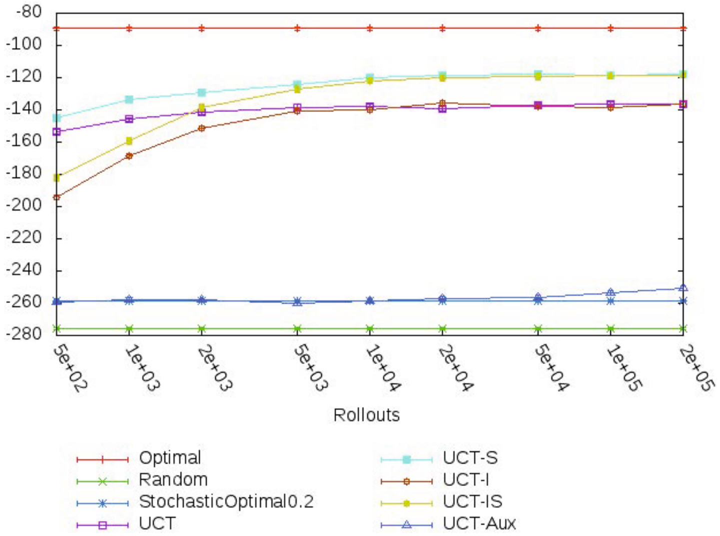


Fig. 8. Bad case of UCT-Aux when coupled with StochasticOptimal.0.2

Table 1. The average number of tree nodes for UCT variants in Obstructed Sailing when coupled with StochasticOptimal.0.2

Samplings	500	1000	2000	5000	10000	20000	50000	100000	200000
UCT	268.4	555.4	1134.6	2694.5	5395.7	11041.7	27107.4	53235.9	107107
UCT-S	268.9	558.6	1157.3	2733.2	5424.7	11212.7	27851.2	54382.2	109308
UCT-I	279.2	583	1200.7	2805	5620	11519.4	28123.8	55393.7	111418
UCT-IS	278.6	586	1209.2	2834.8	5657.1	11704.4	28872.5	56401.2	113682
UCT-Aux	**159.2**	**256.7**	**447.8**	**889.5**	**1341.7**	**1981.3**	**3117.9**	**4317.1**	**5970.11**

UCT-Aux falls into this situation when coupled with suboptimal policies whose estimates are better than random sampling: At every internal node, it artificially creates an arm that is suboptimal but produces preferable reward sequences when compared to other arms with random sampling. As a result, the auxiliary arms are sampled exponentially more often while not necessarily prescribing a good move. Table 1 shows some evidence of this behavior: Given the same number of samplings, UCT-Aux constructs a search tree with significantly less nodes than other variants (up to 20 times). That means many samplings have ended up in non-expansive auxiliary arms because they were preferred.

7.2 Combination of UCT-Aux, UCT-I and UCT-S

UCT-Aux bootstraps UCT in an orthogonal manner to UCT-I and UCT-S, thus allowing combination with these common techniques for further performance boost when many heuristics are available. Figure 9 charts the performance of

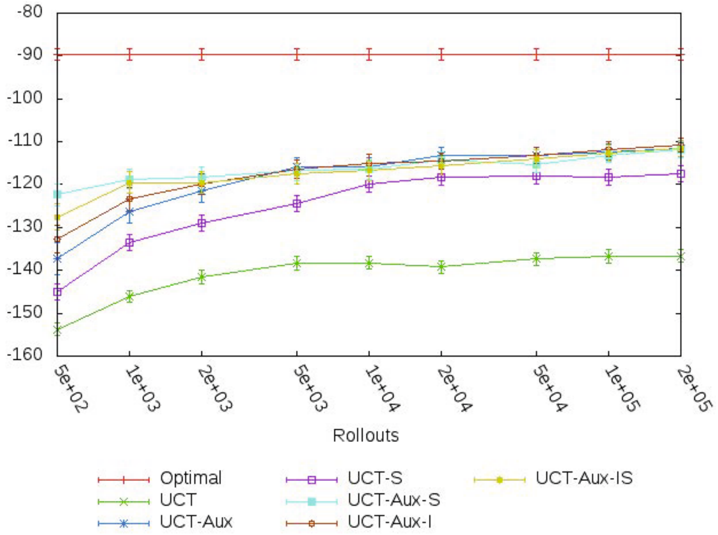


Fig. 9. Combination of UCT-Aux with UCT-I/S/IS in Obstructed Sailing

such combinations in Obstructed Sailing. UCT-Aux variants use SailingTowardsGoal at the auxiliary arms while UCT-I/S variants use StochasticOptimal.0.2 at the ordinary arms. UCT-Aux-S outperforms both UCT-Aux and UCT-S at earlier stage, and matches the better performer among the two, i.e. UCT-Aux, in a long run.

8 Conclusion

In this work, we have introduced a novel yet simple technique to bootstrap UCT with an imperfect heuristic policy in a popular non-adversarial domain, i.e., planning in large-state space MDPs. It is shown to be able to leverage on the well-performing region while avoiding the bad regions of the policy, empirically outperforming other state-of-the-art bootstrapping methods when coupled with the right policy, i.e, the “extreme” kind. Our conclusion is that if such property is known before hand about a certain heuristic, UCT-Aux can be expected to give a real boost over the original UCT, especially in cases with scarce computational resource; otherwise, it would be safer to employ the currently prevalent methods of bootstrapping. As such, a different mentality can be employed when designing heuristics specifically for UCT-Aux: instead of safe heuristics that try to avoid as many flaws as possible, the designer should go for greedier and “riskier” ones. Lastly, since UCT-Aux is orthogonal to other commonly known enhancements, it is a flexible tool that can be combined with others, facilitating more options when incorporating domain knowledge into the vanilla MCTS algorithm. In the future, we plan to examine how to adapt the method to adversarial domains.

Acknowledgments. This research is partially supported by a GAMBIT grant "Tools for Creating Intelligent Game Agents", no. R-252-000-398-490 from the Media Development Authority and an Academic Research Grant no. T1 251RES1005 from the Ministry of Education in Singapore.

References

1. Coulom, R.: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In: van den Herik, H.J., Ciancarini, P., Donkers, H.H.L.M(J.) (eds.) CG 2006. LNCS, vol. 4630, pp. 72–83. Springer, Heidelberg (2007)
2. Kocsis, L., Szepesvári, C.: Bandit Based Monte-Carlo Planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
3. Gelly, S., Silver, D.: Combining online and offline knowledge in UCT. In: ICML 2007: Proceedings of the 24th International Conference on Machine Learning, pp. 273–280. ACM, New York (2007)
4. Chaslot, G., Fiter, C., Hoock, J.-B., Rimmel, A., Teytaud, O.: Adding Expert Knowledge and Exploration in Monte-Carlo Tree Search. In: van den Herik, H.J., Spronck, P. (eds.) ACG 2009. LNCS, vol. 6048, pp. 1–13. Springer, Heidelberg (2010)
5. Finnsson, H., Björnsson, Y.: Simulation-based approach to General Game Playing. In: AAAI 2008: Proceedings of the 23rd National Conference on Artificial Intelligence, pp. 259–264. AAAI Press (2008)
6. Balla, R.K., Fern, A.: UCT for tactical assault planning in real-time strategy games. In: 21st International Joint Conference on Artificial Intelligence, pp. 40–45 (2009)
7. Bouzy, B., Helmstetter, B.: Monte-Carlo Go developments. In: Advances in Computer Games, vol. 10, pp. 159–174 (2004)
8. Coquelin, P.A., Munos, R.: Bandit algorithms for tree search. In: Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, pp. 67–74 (2007)
9. Bellman, R.: A Markovian Decision Process. Indiana Univ. Math. J. 6, 679–684 (1957)
10. Kearns, M., Mansour, Y., Ng, A.Y.: A sparse sampling algorithm for near-optimal planning in large Markov Decision Processes. Machine Learning 49, 193–208 (2002)
11. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47, 235–256 (2002)
12. Brügmann, B.: Monte Carlo Go. Physics Department, Syracuse University. Tech. Rep. (1993)
13. Vanderbei, R.: Sailing strategies: An application involving stochastics, optimization, and statistics, SOS (1996), http://orfe.princeton.edu/~rvdb/sail/sail.html
14. Nguyen, T.H.D., Hsu, D., Lee, W.S., Leong, T.Y., Kaelbling, L.P., Lozano-Perez, T., Grant, A.H.: Capir: Collaborative action planning with intention recognition. In: Proceedings of the Seventh Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE 2011), AAAI. AAAI Press (2011)

Fast Reinforcement Learning with Large Action Sets Using Error-Correcting Output Codes for MDP Factorization

Gabriel Dulac-Arnold[1], Ludovic Denoyer[1],
Philippe Preux[2], and Patrick Gallinari[1]

[1] LIP6–UPMC, Case 169 4 Place Jussieu,
Paris 75005, France
firstname.lastname@lip6.fr

[2] LIFL (UMR CNRS) & INRIA Lille Nord-Europe – Université de Lille
Villeneuve d'Ascq, France
firstname.lastname@inria.fr

Abstract. The use of Reinforcement Learning in real-world scenarios is strongly limited by issues of scale. Most RL learning algorithms are unable to deal with problems composed of hundreds or sometimes even dozens of possible actions, and therefore cannot be applied to many real-world problems. We consider the RL problem in the supervised classification framework where the optimal policy is obtained through a multiclass classifier, the set of classes being the set of actions of the problem. We introduce error-correcting output codes (ECOCs) in this setting and propose two new methods for reducing complexity when using rollouts-based approaches. The first method consists in using an ECOC-based classifier as the multiclass classifier, reducing the learning complexity from $\mathcal{O}(A^2)$ to $\mathcal{O}(A\log(A))$. We then propose a novel method that profits from the ECOC's coding dictionary to split the initial MDP into $\mathcal{O}(\log(A))$ separate two-action MDPs. This second method reduces learning complexity even further, from $\mathcal{O}(A^2)$ to $\mathcal{O}(\log(A))$, thus rendering problems with large action sets tractable. We finish by experimentally demonstrating the advantages of our approach on a set of benchmark problems, both in speed and performance.

1 Introduction

The goal of Reinforcement Learning (RL) and more generally sequential decision making is to learn an optimal policy for performing a certain task within an environment, modeled by a Markov Decision Process (MDP). In RL, the dynamics of the environment are considered as unknown. This means that to obtain an optimal policy, the learner interacts with its environment, observing the outcomes of the actions it performs. Though well understood from a theoretical point of view, RL still faces many practical issues related to the complexity of the environment, in particular when dealing with large state or action sets. Currently, using function approximation to better represent and generalize over

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 180–194, 2012.

the environment is a common approach for dealing with large *state* sets. However, learning with large *action* sets has been less explored and remains a key challenge.

When the number of possible actions A is neither on the scale of 'a few' nor outright continuous, the situation becomes difficult. In particular cases where the action space is continuous (or nearly so), a regularity assumption can be made on the consequences of the actions concerning either a certain smoothness or Lipschitz property over the action space [1, 2, 3]. However, situations abound in which the set of actions is discrete, but the number of actions lies somewhere between 10 and 10^4 (or greater) — Go, Chess, and planning problems are among these. In the common case where the action space shows no regularity *a priori*, it is not possible to make any assumptions regarding the consequence of an action that has never been applied.

In this article, we present an algorithm which can intelligently sub-sample even completely irregular action spaces. Drawing from ideas used in multiclass supervised learning, **we introduce a novel way to significantly reduce the complexity of learning (and acting) with large action sets**. By assigning a multi-bit code to each action, we create binary clusters of actions through the use of *Error Correcting Output Codes* (ECOCs) [4]. Our approach is anchored in Rollout Classification Policy Iteration (RCPI) [5], an algorithm well know for its efficiency on real-world problems. We begin by proposing a simple way to reduce the computational cost of any policy by leveraging the clusters of actions defined by the ECOCs. We then extend this idea to the problem of learning, and propose a new RL method that allows one to find an approximated optimal policy by solving a set of 2-action MDPs. While our first model — ECOC-extended RCPI (ERCPI) — reduces the overall learning complexity from $\mathcal{O}(A^2)$ to $\mathcal{O}(A\log(A))$, our second method — Binary-RCPI (BRCPI) — reduces this complexity even further, to $\mathcal{O}(\log(A))$.

The paper is organized as follow: We give a brief overview of notation and RL in Section 2.1, then introduce RCPI and ECOCs in Sections 2.2 and 2.3 respectively. We present the general idea of our work in Section 3. We show how RCPI can be extended using ECOCs in 3.2, and then explain in detail how an MDP can be factorized to accelerate RCPI during the learning phase in 3.3. An in-depth complexity analysis of the different algorithms is given in Section 3.4. Experimental results are provided on two problems in Section 4. Related work is presented in Section 5.

2 Background

In this section, we cover the three key elements to understanding our work: Markov Decision Problems, Rollout Classification Policy Iteration, and Error-Correcting Output Codes.

2.1 Markov Decision Process

Let a Markov Decision Process $\mathcal{M}$ be defined by a 4-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R)$.

- $\mathcal{S}$ is the set of possible states of the MDP, where $s \in \mathcal{S}$ denotes one state of the MDP.
- $\mathcal{A}$ is the set of possible actions, where $a \in \mathcal{A}$ denotes one action of the MDP.
- $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the MDP's transition function, and defines the probability of going from state s to state s' having chosen action a: $T(s', s, a) = P(s'|s, a)$.
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function defining the expected immediate reward of taking action a in state s. The actual immediate reward for a particular transition is denoted by r.

In this article, we assume that the set of possible actions is the same for all states, but our work is not restricted to this situation; the set of actions can vary with the state without any drawbacks.

Let us define a policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, providing a mapping from states to actions in the MDP. In this paper, without loss of generality, we consider that the objective to fulfill is the optimization of the expected sum of γ-discounted rewards from a given set of states D: $J_\pi(s) = \mathbb{E}[\sum_{k \geq 0} \gamma^k r_{t+k} | s_t = s \in D, \pi]$.

A policy's performance is measured w.r.t. the objective function J_π. The goal of RL is to find an optimal policy π^* that maximizes the objective function: $\pi^* = argmax_\pi J_\pi$.

In an RL problem, the agent knows both $\mathcal{S}$ and $\mathcal{A}$, but is not given the environment's dynamics defined by T and R. In the case of our problems, we assume that the agent may start from any state in the MDP, and can run as many simulations as necessary until it has learned a good policy.

2.2 Rollout Classification Policy Iteration

We anchor our contribution in the framework provided by RCPI [5]. RCPI belongs to the family of Approximate Policy Iteration (API) algorithms, iteratively improving estimates of the Q-function — $Q_\pi(s, a) = \mathbb{E}[J_\pi(s)|\pi]$. In general, API uses a policy π to estimate Q through simulation, and then approximates it by some form of regression on the estimated values, providing $\tilde{Q}_\pi$. This is done first with an initial (and often random) policy π_0, and is iteratively repeated until $\tilde{Q}_\pi$ is properly estimated. $\tilde{Q}_\pi(s, a)$ is estimated by running K *rollouts* i.e. Monte-Carlo simulations using π to estimate the expected reward. The new policy π' is thus the policy that chooses the action with the highest $\tilde{Q}_\pi$-value for each state.

In the case of RCPI, instead of using a function approximator to estimate $\tilde{Q}_\pi$, the best action for a given s is selected using a classifier, without explicitly approximating the Q-value. This estimation is usually done using a binary classifier f_θ over the state-action space such that the new policy can be written as:

$$\pi'(s) = \underset{a \in \mathcal{A}_s}{\operatorname{argmax}} f_\theta(s, a). \tag{1}$$

The classifier's training set $\mathcal{S}_{\mathrm{T}}$ is generated through Monte-Carlo sampling of the MDP, estimating the optimal action for each state sampled. Once generated, these optimal state-action pairs (s, a) are used to train a supervised classifier; the state is interpreted as the feature vector, and the action a as the state's label. In other words, RCPI is an API algorithm that uses Monte Carlo simulations to transform the RL problem into a multiclass classification problem.

2.3 Error-Correcting Output Codes

In the domain of multiclass supervised classification in large label spaces, ECOCs have been in use for a while [4]. We will cover ECOCs very briefly here, as their adaptation to an MDP formalism is well detailed in Section 3.2.

Given a multiclass classification task with a label set $\mathcal{Y}$, the $|\mathcal{Y}|$ class labels can be encoded as binary integers using as few as $C = \log_2(|\mathcal{Y}|)$ bits. ECOCs for classification assume that each label is associated to a binary code of length[1] $C = \gamma \log(|\mathcal{Y}|)$ with $\gamma \geq 1$.

The main principle of multiclass classifiers with ECOCs is to learn to predict the output code instead of directly predicting the label, transforming a supervised learning problem with $|\mathcal{Y}|$ classes into a set of $C = \gamma \log(|\mathcal{Y}|)$ binary supervised learning problems. Once trained, the class of a datum x can be inferred by passing the datum to all the classifiers and concatenating their output into a predicted label code: $code(x) = (f_{\theta_0}(x), \cdots, f_{\theta_C}(x))$. The predicted label is thus the label with the closest code in terms of Hamming distance. As a side note, Hamming distance look-ups can be done in logarithmic time by using tree-based approaches such as k-d trees [6]. ECOCs for classification can thus infer with a complexity of $\mathcal{O}(log(|\mathcal{Y}|)$.

3 Extended and Binary RCPI

We separate this paper's contributions into two parts, the second part building on the first one. We begin by showing how ECOCs can be easily integrated into a classifier-based policy, and proceed to show how the ECOC's coding matrix can be used to factorize RCPI into a much less complex learning algorithm.

3.1 General Idea

The general idea of our two algorithms revolves around the use of ECOCs for representing the set of possible actions, $\mathcal{A}$. This approach assigns a multi-bit code of length $C = \gamma \log(A)$ to each of the A actions. The codes are organized in a *coding matrix*, illustrated in Figure 1 and denoted $\mathbf{M}^c$. Each row corresponds to one action's binary code, while each column is a particular dichotomy of the action space corresponding to that column's associated bit b_i. In effect, each

[1] Different methods exist for generating such codes. In practice, it is customary to use redundant codes where $\gamma \approx 10$.

	b_1	b_2	b_3
a_1	$+$	$+$	$-$
a_2	$-$	$+$	$-$
a_3	$+$	$-$	$+$
a_4	$-$	$+$	$+$
a_5	$+$	$-$	$-$

Fig. 1. An example of a 5-actions, $C = 3$-bits coding matrix. The code of action 1 is $(+, +, -)$.

column is a projection of the A-dimensional action space into a 2-dimensional binary space. We denote $\mathbf{M}^c_{[a,*]}$ as the a^{th} row of $\mathbf{M}^c$, which corresponds to a's binary code. $\mathbf{M}^c_{[a,i]}$ corresponds to bit b_i of action a's binary code.

Our main idea is to consider that each bit corresponds to a binary sub-policy denoted π_i. By combining these sub-policies, we can derive the original policy π one wants to learn as such:

$$\pi(s) = \operatorname*{argmin}_{a \in \mathcal{A}} d_{\mathrm{H}}(\mathbf{M}^c_{[a,*]}, (\pi_1(s), \cdots, \pi_C(s)), \qquad (2)$$

where $\mathbf{M}^c_{[a,*]}$ is the binary code for action a, and d_{H} is the Hamming distance. For a given a state s, each sub-policy provides a binary action $\pi_i(s) \in \{-, +\}$, thus producing a binary vector of length C. $\pi(s)$ chooses the action a with the binary code that has the smallest Hamming distance to the concatenated output of the C binary policies.

We propose two variants of RCPI that differ by the way they learn these sub-policies. ECOC-extended RCPI (ERCPI) replaces the standard definition of π by the definition in Eq. (2), both for learning and action selection. The Binary-RCPI method (BRCPI) relaxes the learning problem and considers that all the sub-policies can be learned independently on separate binary-actioned MDPs, resulting in a very rapid learning algorithm.

3.2 ECOC-Extended RCPI

ERCPI takes advantage of the policy definition in Equation (2) to decrease RCPI's complexity. The C sub-policies — $\pi_{i \in [1,C]}$ — are learned simultaneously on the original MDP, by extending the RCPI algorithm with an ECOC-encoding step, as described in Algorithm 1. As any policy improvement algorithm, ERCPI iteratively performs the following two steps:

Simulation Step. This consists in performing Monte Carlo simulations to estimate the quality of a set of state-action pairs. From these simulations, a set of training examples $\mathcal{S}_{\mathrm{T}}$ is generated, in which data are states, and labels are the estimated best action for each state.

Learning Step. For each bit b_i, $\mathcal{S}_{\mathrm{T}}$ is used to create a binary label training set $\mathcal{S}_{\mathrm{T}}^i$. Each $\mathcal{S}_{\mathrm{T}}^i$ is then used to train a classifier f_{θ_i}, providing sub-policy π'_i as in Eq. (1). Finally, the set of C sub-policies are combined to provide the final improved policy as in Eq. (2).

ERCPI's training algorithm is presented in Alg. 1.

The **Rollout** function used by ERCPI is identical to the one used by RCPI — π is used to estimate a certain state-action tuple's expected reward, $\tilde{Q}_\pi(s,a)$.

Algorithm 1. ERCPI

```
   Data:
   $\mathcal{S}_R$: uniformly sampled state set; $\mathcal{M}$: MDP; $\pi_0$: initial policy; $K$: number of
   trajectories; $T$: maximum trajectory length
 1 $\pi = \pi_0$
 2 repeat
 3     $\mathcal{S}_{\mathrm{T}} = \emptyset$
 4     foreach $s \in \mathcal{S}_R$ do
 5         foreach $a \in A$ do
 6             $\tilde{Q}_\pi(s,a) \leftarrow$ Rollout$(\mathcal{M}, s, a, K, \pi)$
 7         end
 8         $\mathcal{A}^* = \mathrm{argmax}_{a \in \mathcal{A}} \tilde{Q}_\pi(s,a)$
 9         foreach $a^* \in \mathcal{A}^*$ do
10             $\mathcal{S}_{\mathrm{T}} = \mathcal{S}_{\mathrm{T}} \cup \{(s,a^*)\}$
11         end
12     end
13     foreach $i \in [1,C]$ do
14         $\mathcal{S}_{\mathrm{T}}^i = \emptyset$
15         foreach $(s,a) \in \mathcal{S}_T$ do
16             $a_i = \mathbf{M}^c_{[a,i]}$
17             $\mathcal{S}_{\mathrm{T}}^i = \mathcal{S}_{\mathrm{T}}^i \cup (s, a_i)$
18         end
19         $f_{\theta_i} =$ Train$(\mathcal{S}_{\mathrm{T}}^i)$
20         $\pi'_i$ from $f_{\theta_i}$ as defined in Eq. (1)
21     end
22     $\pi'$ as defined in Eq. (2)
23     $\pi = \alpha(\pi, \pi')$
24 until $\pi \sim \pi'$;
25 return $\pi$
```

Up to line 12 of Algorithm 1, ERCPI is in fact algorithmically identical to RCPI, with the slight distinction that only the best (s,a^*) tuples are kept, as is usual when using RCPI with a multiclass classifier.

ERCPI's main difference appears starting line 13; it is here that the original training set $\mathcal{S}_{\mathrm{T}}$ is mapped onto the C binary action spaces, and that each individual sub-policy π_i is learned. Line 16 replaces the original label of state s by its binary label in π_i's action space — this corresponds to bit b_i of action a's code.

The **Train** function on line 19 corresponds to the training of π_i's corresponding binary classifier on $\mathcal{S}_{\mathrm{T}}^i$. After this step, the global policy π' is defined according to Eq.(2). Note that, to ensure the stability of the algorithm, the new policy π obtained after one iteration of the algorithm is an alpha-mixture policy between the old π and the new π' obtained by the classifier (cf. line 23).

3.3 Binarized RCPI

ERCPI splits the policy improvement problem into C individual problems, but training still needs π, thus requiring the full set of binary policies. Additonnally, for each state, all A actions have to be evaluated by Monte Carlo simulation (Alg. 1, line 5). To reduce the complexity of this algorithm, we propose learning the C binary sub-policies — $\pi_{i\in[1,C]}$ — **independently**, transforming our initial MDP into C sub-MDPs, each one corresponding to the environment in which a particular π_i is acting.

Each of the π_i binary policies is dealing with its own particular representation of the action space, defined by its corresponding column in $\mathbf{M}^c$. For training, best-action selections must be mapped into this binary space, and each of the π_i's choices must be combined to be applied back in the original state space.

Let $\mathcal{A}_i^+, \mathcal{A}_i^- \subset \mathcal{A}$ be the action sets associated to π_i such that:

$$\begin{aligned} \mathcal{A}_i^+ &= \{a \in \mathcal{A} \mid \mathbf{M}^c_{[a,i]} = `+'\} \\ \mathcal{A}_i^- &= \mathcal{A} \setminus \mathcal{A}_i^+ = \{a \in \mathcal{A} \mid \mathbf{M}^c_{[a,i]} = `-'\}. \end{aligned} \tag{3}$$

For a particular i, $\mathcal{A}_i^+$ is the set of original actions corresponding to sub-action $+$, and $\mathcal{A}_i^-$ is the set of original actions corresponding to sub-action $-$.

We can now define C new binary MDPs that we name sub-MDPs, and denote $\mathcal{M}_{i\in[1,C]}$. They are defined from the original MDP as follows:

- $\mathcal{S}_i = \mathcal{S}$, the same state-set as the original MDP.
- $\mathcal{A}_i = \{+, -\}$.
- $T_i = T(s', s, a)P(a|a_i) = P(s'|s, a)P(a|a_i)$, where $P(a|a_i)$ is the probability of choosing action $a \in \mathcal{A}^{a_i}$, knowing that the sub-action applied on the sub-MDP $\mathcal{M}_i$ is $a_i \in \{+, -\}$. We consider $P(a|+)$ to be uniform for $a \in \mathcal{A}^+$ and null for $a \in \mathcal{A}^-$, and vice versa. $P(s'|s, a)$ is the original MDP's transition probability.
- $R_i(s, a_i) = \sum_{a \in \mathcal{A}_i^{a_i}} P(a|a_i)R(s, a)$.

Each of these new MDPs represents the environment in which a particular binary policy π_i operates. We can consider each of these MDPs to be a separate RL problem for its corresponding binary policy.

In light of this, we propose to transform RCPI's training process for the base MDP into C new training processes, each one trying to find an optimal π_i for its corresponding $\mathcal{M}_i$. Once all of these binary policies have been trained, they can be used during inference in the manner described in Section 3.2.

The main advantage of this approach is that, since each of the $\gamma \log(A)$ sub-problems in Algorithm 2 is modeled as a binary-actioned MDP, increasing the number of actions in the original problem simply increases the number of sub-problems logarithmically, without increasing the complexity of these sub-problems – see Section 3.4.

Algorithm 2. BRCPI

```
   Data:
   $\mathcal{S}_R$: uniformly sampled state set; $\mathcal{M}$: MDP; $\pi_0$: random policy; $K$: number of
   trajectories; $T$: maximum trajectory length; $C$: number of binary MDPs
 1 foreach $i \in C$ do
 2     $\pi_i = \pi_0$
 3     repeat
 4         $\mathcal{S}_{\mathrm{T}} = \emptyset$
 5         foreach $s \in \mathcal{S}_R$ do
 6             foreach $a \in \{+, -\}$ do
 7                 $\tilde{Q}_\pi(s,a) \leftarrow$ Rollout($\mathcal{M}_i, s, a, K, \pi_i$)
 8             end
 9             $\mathcal{A}^* = \mathrm{argmax}_{a \in \mathcal{A}}\, \tilde{Q}_\pi(s,a)$
10             foreach $a^* \in \mathcal{A}^*$ do
11                 $\mathcal{S}_{\mathrm{T}} = \mathcal{S}_{\mathrm{T}} \cup \{(s, a^*)\}$
12             end
13         end
14         $f_{\theta_i}$ = Train($\mathcal{S}_{\mathrm{T}}$)
15         $\pi'_i$ from $f_{\theta_i}$ as defined in Eq. (1)
16         $\pi_i = \alpha(\pi_i, \pi'_i)$
17     until $\pi_i \sim \pi'_i$;
18     return $\pi$ as defined in Eq. (2)
19 end
```

Let us now discuss some details of BRCPI, as described in Algorithm 2. BRCPI resembles RCPI very strongly, except that instead of looping over the A actions on line 6, BRCPI is only sampling $\tilde{Q}$ for $+$ or $-$ actions. However, the inner loop is run $C = \gamma \log(A)$ times, as can be seen on line 1 of Algorithm 2.

Within the **Rollout** function (line 7), if π_i chooses sub-action '+', an action a_i from the original MDP is sampled from $\mathcal{A}_i^+$ following $P(a|a_i)$, and the MDP's transition function is called using this action. This effectively estimates the expected reward of choosing action $+$ in state s.

As we saw in Section 3.2, each $\mathcal{A}_i$ is a different binary projection of the original action set. Each of the π_i classifiers is thus making a decision considering a different split of the action space. Some splits may make no particular sense w.r.t. to the MDP at hand, and therefore the expected return of that particular π_i's $\mathcal{A}_i^+$ and $\mathcal{A}_i^-$ may be equal. This does not pose a problem, as that particular sub-policy will simply output noise, which will be corrected for by more pertinent splits given to the other sub-policies.

3.4 Computational Cost and Complexity

We study the computational cost of the proposed algorithms in comparison with the RCPI approach and present their respective complexities.

In order to define this cost, let us consider that $\mathcal{C}(S, A)$ is the time spent learning a multiclass classifier on S examples with A possible outputs, and $\mathcal{I}(A)$ is the cost of classifying one input.

The computational cost of one iteration of RCPI or ERCPI is composed of both a **simulation cost** — which corresponds to the time spent making Monte Carlo Simulation using the current policy — and a **learning cost** which corresponds to the time spent learning the classifier that will define the next policy[2]. This cost takes the following general form:

$$Cost = SAK \times T\mathcal{I}(A) + \mathcal{C}(S, A), \tag{4}$$

where $T\mathcal{I}(A)$ is the cost of sampling one trajectory of size T, $SAK \times T\mathcal{I}(A)$ is the cost of executing the K Monte Carlo Simulations over S states testing A possible actions, and $\mathcal{C}(S, A)$ is the cost of learning the corresponding classifier[3].

The main difference between RCPI and ERCPI comes from the values of $\mathcal{I}(A)$ and $\mathcal{C}(S, A)$. When comparing ERCPI with a RCPI algorithm using a *one-vs-all* (RCPI-OVA) multiclass classifier — one binary classifier learned for each possible action — it is easy to see that our method reduces both $\mathcal{I}(A)$ and $\mathcal{C}(S, A)$ by a factor of $\frac{A}{\log A}$ — cf. Table 1.

Table 1. Cost of one iteration of RCPI OVA, ERCPI, and BRCPI. S is the number of states, A the number of actions, K the number of rollouts, T is trajectory length, $\mathcal{C}(S, A)$ is the cost of learning a classifier for S states, A actions

Algorithm	Simulation Cost	Learning Cost
RCPI-OVA	$SAK(TA)$	$A.\mathcal{C}(S)$
ERCPI	$SAK(T\gamma\log(A))$	$\gamma\log(A).\mathcal{C}(S)$
BRCPI	$\gamma\log(A)\ (2SK(2T))$	$\gamma\log(A)\mathcal{C}(S)$

Table 2. Complexity w.r.t. the number of possible actions

Method	RCPI OVA	ERCPI	BRCPI
Complexity	$\mathcal{O}(A^2)$	$\mathcal{O}(A\log(A))$	$\mathcal{O}(\log(A))$

When considering the BRCPI algorithm, $\mathcal{I}$ and $\mathcal{C}$ are reduced as in ERCPI. However, the simulation cost is reduced as well, as our method proposes to learn a set of optimal binary policies on $\gamma\log(A)$ binary sub-MDPs. For each of these sub-problems, the simulation cost is $2SK(2T)$ since the number of possible actions is only 2. The learning cost corresponds to learning only $\gamma\log(A)$ binary

[2] In practice, when there are many actions, simulation cost is significantly higher than learning cost, which is thus ignored [7].

[3] We do not consider the computational cost of transitions in the MDP.

classifiers resulting in a very low cost — cf. Table 1. The overall resulting complexity w.r.t. to the number of actions is presented in Table 2, showing that the complexity of BRCPI is only logarithmic. In addition, it is important to note that each of the BRCPI sub-problems is atomic, and are therefore easily parallelized. To illustrate these complexities, computation times are reported in the experimental section.

4 Experiments

In this paper, our concern is really about being able to deal with a large number of uncorrelated actions in practice. Hence, the best demonstration of this ability is to provide an experimental assessment of ERCPI and BRCPI. In this section, we show that BRCPI exhibits very important speed-ups, turning days of computations into hours or less.

4.1 Protocol

We evaluate our approaches on two baseline RL problems: Mountain Car and Maze.

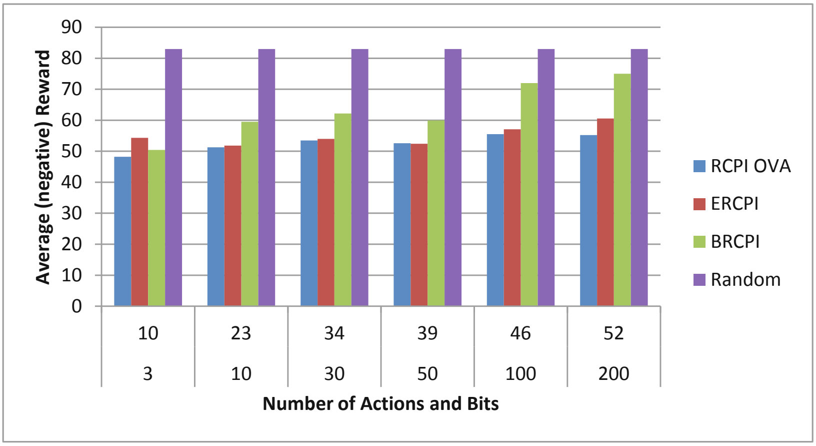


Fig. 2. Mountain Car: Average reward (negative value: the smaller, the better) obtained by the different algorithms on 3 runs with different numbers of actions. On the X-axis, the first line corresponds to $\gamma \log(A)$ while the second line is the number of actions A.

The first problem, **Mountain Car**, is well-known in the RL community. Its definition varies, but it is usually based on a discrete and small set of actions (2 or 3). However, the actions may be defined over a continuous domain, which is more “realistic”. In our experiment, we discretize the range of accelerations to obtain a discrete set of actions. Discretization ranges from coarse to fine in the experiments, thus allowing us to study the effect of the size of the action set on the performance of our algorithms. The continuous *state* space is handled by way of tiling [8]. The reward at each step is -1, and each episode has a maximum length of 100 steps. The overall reward thus measures the ability of the obtained policy to push the car up to the mountain quickly.

The second problem, **Maze**, is a 50x50 grid-world problem in which the learner has to go from the left side to the right side of a grid. Each cell of the grid corresponds to a particular negative reward, either -1, -10, or -100. For the simplest case, the agent can choose either to move *up*, *down*, or *right*, resulting in a 3-action MDP. We construct more complex action sets by generating all sequences of actions of a defined length i.e. for length 2, the 6 possible actions are *up-up*, *up-right*, *down-up*, etc. Contrary to Mountain Car, there is no notion of similarity between actions in this maze problem w.r.t. their consequences. Each state is represented by a vector of features that contains the information about the different types of cells that are contained in a 5x5 grid around the agent. The overall reward obtained by the agent corresponds to its ability to go from the left to the right, avoiding cells with high negative rewards.

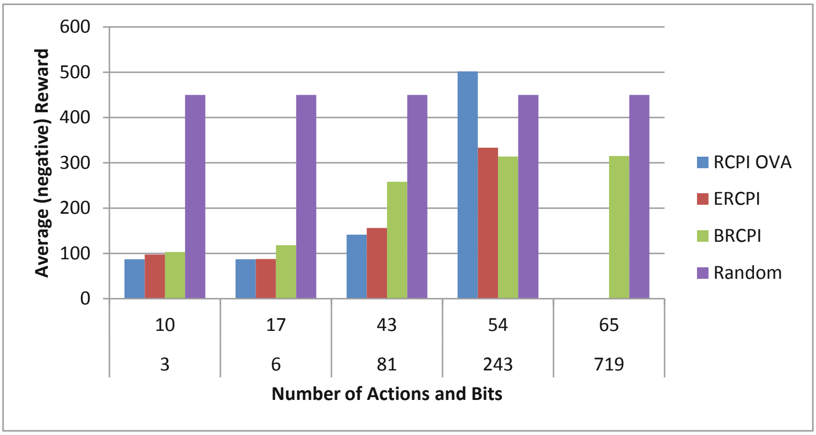


Fig. 3. Maze: Average reward (negative value: the smaller, the better) obtained by the different algorithms on 3 different random mazes with different numbers of actions. On the X-axis, the first line corresponds to $\gamma \log(A)$ while the second line is the number of actions A. OVA and ERCPI were intractable for 719 actions. Note that for 243 actions, RCPI-OVA learns a particularly bad policy.

In both problems, training and testing states are sampled uniformly in the space of the possible states. We have chosen to sample $S = 1000$ states for each problem, the number of trajectories made for each state-action pair is $K = 10$. The binary base learner is a hinge-loss perceptron learned with 1000 iterations by stochastic gradient-descent algorithm. The error correcting codes have been generated using a classical random procedure as in [9]. The α-value of the alpha-mixture policy is 0.5.

4.2 Results

The average rewards obtained after convergence of the three algorithms are presented in Figures 3 and 2 with a varying number of actions. The average reward of a random policy is also illustrated. First of all, one can see that RCPI-OVA and ERCPI perform similarly on both problems except for Maze with 243 actions. This can be explained by the fact that OVA strategies are not able to deal with problems with many classes when they involve solving binary classification problems with few positive examples. In this setting, ECOC-classifiers are known to perform better. BRCPI achieves lower performances than OVA-RCPI and ERCPI. Indeed, BRCPI learns optimal independent binary policies that, when used together, only correspond to a sub-optimal overall policy. Note that even with a large number of actions, BRCPI is able to learn a relevant policy — in particular, Maze with 719 actions shows BRCPI is clearly better than the random baseline, while the other methods are simply intractable. This is a very interesting result since it implies that BRCPI is able to find non-trivial policies when classical approaches are intractable.

Table 3 provides the computation times for one iteration of the different algorithms for Mountain Car with 100 actions. ERCPI speeds-up RCPI by a factor 1.4 while BRCPI is 12.5 times faster than RCPI, and 23.5 times faster when considering only the simulation cost. This explains why Figure 3 does not show performances obtained by RCPI and ERCPI on the maze problem with 719 actions: in that setting, one iteration of these algorithms takes days while only requiring a few hours with BRCPI. Note that these speedup values increase with the number of actions.

At last, Figure 4 gives the performance of BRCPI depending on the number of rollouts, and shows that a better policy can be found by increasing the value of K. Note that, even if we use a large value of K, BRCPI's running time remains low w.r.t. to OVA-RCPI and ERCPI.

Table 3. Time (in seconds) spent for one iteration — during simulation and learning — of the different variants of the RCPI algorithms using a Xeon-X5690 Processor and a TESLA M2090 GPU for $K = 10$ and $S = 1000$. The total speedup (and simulation speedup) w.r.t. OVA-RCPI are presented on the last column.

Mountain Car - 100 Actions - 46 bits				
	Sim.	Learning	Total	Speedup
OVA	4,312	380	4,698	×1.0
ERCPI	3,188	190	3,378	×1.4(×1.35)
BRCPI	184	190	374	×12.5(×23.5)

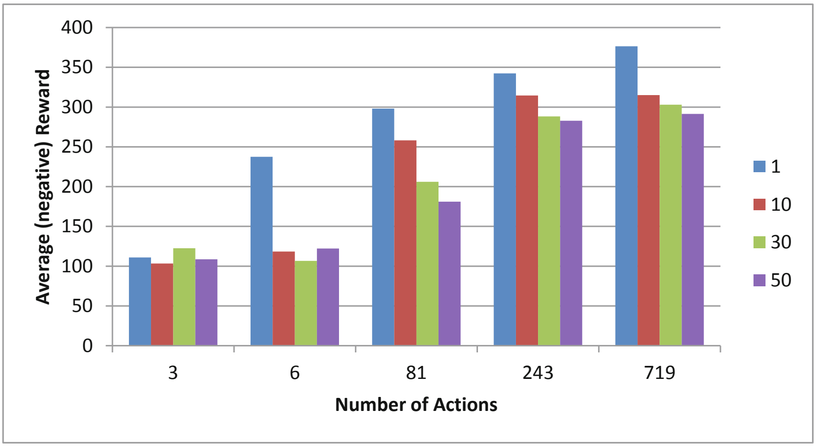


Fig. 4. Maze Rollouts: Average reward (negative value: the smaller, the better) obtained by BRCPI for $K = 1, 10, 30, 50$

5 Related Work

Rollout Classification Policy Iteration [5] provides an algorithm for RL in MDPs that have a very large state space. RCPI's Monte-Carlo sampling phase can be very costly, and a couple approaches have been provided to better sample the state space [10], thus leading to speedups when using RCPI. Recently, the effectiveness of RCPI has been theoretically assessed [7]. The well known efficiency of this method for real-world problems and its inability to deal with many actions have motivated this work.

Reinforcement Learning has long been able to scale to state-spaces with many (if infinite) states by generalizing the value-function over the state space [11, 12]. Tesauro first introduced rollouts [13], leveraging Monte-Carlo sampling for exploring a large state and action space. Dealing with large action spaces has additionally been considered through sampling or gradient descent on Q [3, 1], but these approaches assume a well-behaved Q-function, which is hardly guaranteed.

There is one vein of work reducing action-space look-ups logarithmically by imposing some form of binary search over the action space [14, 15]. These approaches augment the MDP with a structured search over the action space, thus placing the action space's complexity in the state space. Although not inspirational to ERCPI, these approaches are similar in their philosophy. However, neither proposes a solution to speeding up the learning phase as BRCPI does, nor do they eschew value functions by relying solely on classifier-based approaches as ERCPI does.

Error-Correcting Output Codes were first introduced by Dietterich and Bakiri ([4]) for use in the case of multi-class classification. Although not touched upon in this article, coding dictionary construction can be a key element to the ability of the ECOC-based classifier's abilities[16]. Although in our case we rely on randomly generated codes, codes can be learned from the actual training data [17] or from an *a priori* metric upon the classes space or a hierarchy [18].

6 Conclusion

We have proposed two new algorithms which aim at obtaining a good policy while learning faster than the standard RCPI algorithm. ERCPI is based on the use of Error Correcting Output Codes with RCPI, while BRCPI consists in decomposing the original MDP in a set of binary-MDPs which can be learned separately at a very low cost. While ERCPI obtains equivalent or better performances than the classical *One Vs. All* RCPI implementations at a lower computation cost, BRCPI allows one to obtain a sub-optimal policy very fast, even if the number of actions is very large. We believe that there are plenty of high-complexity situations where having a policy that is even slightly better than random can be very advantageous; in the case of ERCPI we can get sub-optimal policies rapidly, which provide at least *some* solution to an otherwise intractable problem. The complexity of the proposed solutions are $\mathcal{O}(A\log(A))$ and $\mathcal{O}(\log(A))$ respectively, in comparison to RCPI's complexity of $\mathcal{O}(A^2)$. Note that one can use BRCPI to discover a good policy, and then ERCPI in order to improve this policy; this practical solution is not studied in this paper.

This work opens many new research perspectives: first, as the performance of BRCPI directly depends on the quality of the codes generated for learning, it can be very interesting to design automatic methods able to find the well-adapted codes, particularly when one has a metric over the set of possible actions. From a theoretical point of view, we plan to study the relation between the performances of the sub-policies π_i in BRCPI and the performance of the final obtained policy π. At last, the fact that our method allows one to deal with problems with thousands of discrete actions also opens many applied perspectives, and can allow us to find good solutions for problems that have never been studied before because of their complexity.

Acknowledgements. This work was partially supported by the French National Agency of Research (Lampada ANR-09-EMER-007). The authors gratefully acknowledge the many discussions with Moustapha Cisse.

References

1. Lazaric, A., Restelli, M., Bonarini, A.: Reinforcement Learning in Continuous Action Spaces through Sequential Monte Carlo Methods. In: Proc. of NIPS 2007 (2007)
2. Bubeck, S., Munos, R., Stoltz, G., Szepesvári, C., et al.: X-armed bandits. Journal of Machine Learning Research 12, 1655–1695 (2011)

3. Negoescu, D., Frazier, P., Powell, W.: The knowledge-gradient algorithm for sequencing experiments in drug discovery. INFORMS J. on Computing 23(3), 346–363 (2011)
4. Dietterich, T., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Jo. of Art. Int. Research 2, 263–286 (1995)
5. Lagoudakis, M.G., Parr, R.: Reinforcement learning as classification: Leveraging modern classifiers. In: Proc. of ICML 2003 (2003)
6. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM 18(9), 509–517 (1975)
7. Lazaric, A., Ghavamzadeh, M., Munos, R.: Analysis of a classification-based policy iteration algorithm. In: Proc. of ICML 2010, pp. 607–614 (2010)
8. Sutton, R.: Generalization in reinforcement learning: Successful examples using sparse coarse coding. In: Proc. of NIPS 1996, pp. 1038–1044 (1996)
9. Berger, A.: Error-correcting output coding for text classification. In: Workshop on Machine Learning for Information Filtering, IJCAI 1999 (1999)
10. Dimitrakakis, C., Lagoudakis, M.G.: Rollout sampling approximate policy iteration. Machine Learning 72(3), 157–171 (2008)
11. Tham, C.: Modular on-line function approximation for scaling up reinforcement learning. PhD thesis, University of Cambridge (1994)
12. Tesauro, G.: Practical issues in temporal difference learning. Machine Learning 8, 257–277 (1992)
13. Tesauro, G., Galperin, G.R.: On-Line Policy Improvement Using Monte-Carlo Search. In: Proc. of NIPS 1997, pp. 1068–1074 (1997)
14. Pazis, J., Lagoudakis, M.G.: Reinforcement Learning in Multidimensional Continuous Action Spaces. In: Proc. of Adaptive Dynamic Programming and Reinf. Learn., pp. 97–104 (2011)
15. Pazis, J., Parr, R.: Generalized Value Functions for Large Action Sets. In: Proc. of ICML 2011, pp. 1185–1192 (2011)
16. Beygelzimer, A., Langford, J., Zadrozny, B.: Machine learning techniques reductions between prediction quality metrics. In: Performance Modeling and Engineering, pp. 3–28 (2008)
17. Crammer, K., Singer, Y.: On the Learnability and Design of Output Codes for Multiclass Problems. Machine Learning 47(2), 201–233 (2002)
18. Cissé, M., Artieres, T., Gallinari, P.: Learning efficient error correcting output codes for large hierarchical multi-class problems. In: Workshop on Large-Scale Hierarchical Classification ECML/PKDD 2011, pp. 37–49 (2011)

Learning Policies for Battery Usage Optimization in Electric Vehicles

Stefano Ermon, Yexiang Xue, Carla Gomes, and Bart Selman

Department of Computer Science
Cornell University
Ithaca, NY
{ermonste,yexiang,gomes,selman}@cs.cornell.edu

Abstract. The high cost, limited capacity, and long recharge time of batteries pose a number of obstacles for the widespread adoption of electric vehicles. Multi-battery systems that combine a standard battery with supercapacitors are currently one of the most promising ways to increase battery lifespan and reduce operating costs. However, their performance crucially depends on how they are designed and operated.

In this paper, we formalize the problem of optimizing real-time energy management of multi-battery systems as a stochastic planning problem, and we propose a novel solution based on a combination of optimization, machine learning and data-mining techniques. We evaluate the performance of our intelligent energy management system on various large datasets of commuter trips crowdsourced in the United States. We show that our policy significantly outperforms the leading algorithms that were previously proposed as part of an open algorithmic challenge.

1 Introduction

Electric vehicles, partially or fully powered by batteries, are one of the most promising directions towards a more sustainable transportation system. However, the high costs, limited capacities, and long recharge times of batteries pose a number of obstacles for their widespread adoption. Several researchers in the field of Computational Sustainability [1] have addressed aspects of this problem. In particular, there is an active line of research focusing on improving navigation systems with novel routing algorithms, both by taking into account specific features of electric vehicles [2], and by considering new aspects such as real-time information about road conditions and traffic lights [3].

In this paper, we focus on a complementary aspect of the problem that is optimizing the energy efficiency of batteries in electric vehicles. There are two main sources of inefficiencies in batteries. The first one is that due to internal resistance, battery energy is partially wasted as heat when it is charged and discharged. The second one is that due to Peukert's Law, the actual delivered capacity of a battery depends on the rate at which it is discharged. Furthermore, current battery technology imposes rather severe limits on the number of charge/recharge cycles a battery can handle, thus reducing their lifespan and increasing operating costs.

One promising direction towards addressing these issues are multi-battery systems, such as the ones proposed in [4] and [5], which integrate a standard battery with one or

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 195–210, 2012.

more supercapacitors, as depicted in Figure 1. Intuitively, the idea is that while the battery is good at holding the charge for long times, the supercapacitor is efficient for rapid cycles of charge and discharge. Using the capacitor as an energy buffer, one can significantly increase the battery's lifespan by reducing its duty. In fact, although supercapacitors have low energy densities, they behave like an ideal battery that can efficiently handle millions of full charge/discharge cycles. The performance of these systems is heavily dependent on how they are managed. In this direction, there has been recent work in the automated planning community on the optimal scheduling of multi-battery systems [6, 7]. However, previous work assumes full knowledge of an underlying probabilistic model describing the system, which is not available for electric vehicles. Since it would be very difficult to construct such a model using a priori information, we take a data driven approach to the problem. In particular, we leverage a large dataset of commuter trips collected across the United States by Chargecar [8], a crowdsourcing project open to the public, and we construct an efficient management scheme using a sample-based optimization approach. Specifically, after defining a suitable set of features, we learn an empirical Markov Decision Process (MDP) model from the available data, and we compute a policy that optimizes the average performance. This policy is represented as a large table of state-action pairs, and is only defined for states that were previously observed in the dataset, while we wish to construct a more general management scheme that applies to a wider range of scenarios. We therefore use this policy as a training set, and we use supervised learning techniques to learn a new policy that compactly represents the information available and generalizes to situations previously unseen in the training set. This policy is shown to outperform the leading algorithms that were previously proposed as part of an open algorithmic challenge.

2 Sampling-Based Optimization

We consider a probabilistic planning problem formulated as a Markov Decision Process (MDP). An MDP is a tuple (S, A, P, c) where S is a set of states, A is a set of actions, P is a set of transition probabilities and $c : S \times A \times S \mapsto \mathbb{R}$ is an (immediate) cost function. If an agent executes an action $a \in A$ while in a state $s \in S$, then it incurs in an immediate cost $c(s, a, s')$ and it transitions to a new state $s' \in S$ with probability $P(s'|s, a)$. We denote by $\mathcal{A}(s) \subseteq A$ the set of actions available while the agent is in state s. Further, there exists a finite set of *goal states* $G \subseteq S$, where the agent stops to execute actions, and no longer incurs in any cost.

In this paper, we consider a class of factored MDPs where $S = X \times Y$ so that any state $s \in S$ has two components $s = (x, y)$ with $x \in X$ and $y \in Y$. We assume the dynamics of the two components are independent, i.e. the transition probabilities can be factored as follows

$$P((x', y')|(x, y), a) = P_x(x'|x)P_y(y'|y, a).$$

Notice that x' does not depend on the action a, which only affects the component y'. The two components of the state are however coupled by the immediate cost function $c((x, y), a, (x', y'))$, which can depend on both x and y. We assume that $P_y(\cdot)$ and the immediate cost function $c(\cdot)$ are known but $P_x(\cdot)$ is unknown. However, we are given a set of K i.i.d. sample trajectories $\mathcal{T}_1, \cdots, \mathcal{T}_K$ of the x component of the state space, where

$$\mathcal{T}_i = (x_0^i, x_1^i, \cdots, x_{T_i-1}^i)$$

is sampled according to $P_x(x'|x)$ and $x_{T_i-1}^i \in G_x$ is a goal state. For example, in our battery management application, each trajectory corresponds to one commuter trip. Given this information, our objective is to find an admissible policy that minimizes the expected cost for this partially unknown MDP.

Given $x, x' \in X$, let $f(x, x')$ be the empirical transition probability from x to x' according to the available samples (the number of times x' appears immediately after x over the number of times x appears). We can define DP equations based on the sampled transition probabilities as follows

$$V(x,y) = \min_{a \in \mathcal{A}(x,y)} \left(\sum_{x' \in X} \sum_{y' \in Y} f(x,x') P_y(y'|y,a) \left(c((x,y), a, (x',y')) + V(x',y') \right) \right)$$

for all observed states $x \in \bigcup \mathcal{T}_i$. Solving the DP equations, we can compute the "optimal posterior action" $a^*(s) = a^*(x,y)$ for all $x \in \mathcal{T}_i$ and for all $y \in Y$, that is the action minimizing the total expected cost according to our maximum-likelihood estimate of the underlying MDP model. Notice that the "optimal posterior action" $a^*(s)$ converges to the true optimal action for the MDP as $K \to \infty$ because $f(x,x') \to P_x(x'|x)$ (assuming the initial states x_0^i are uniformly sampled).

Although the number of distinct states $x \in \bigcup \mathcal{T}_i \subseteq X$ can be very large, the samples $\mathcal{T}_1, \cdots, \mathcal{T}_K$ do not necessarily cover the entire state space X. We therefore wish to obtain a compact representation of the policy $a^*(\cdot)$, that hopefully will be able to generalize to states $x \in X$ such that $x \notin \bigcup \mathcal{T}_i$, i.e. states previously unseen in the set of available samples. We therefore generate a labeled training set of state-action pairs

$$\bigcup_i \{((x,y), a^*(s)), x \in \mathcal{T}_i, y \in Y\}$$

and we use supervised learning to learn a policy $\pi : S \to A$. Notice that the particular structure of the problem, with independent dynamics and partially known transition probabilities, allows us to artificially generate $|Y|$ time more training examples than what we originally started with. This aspect leads to significant improvements in our battery management application problem.

2.1 Related Work

Sampling-based solutions, where a finite number of sampled trajectories is used to optimize performance, are a popular technique in the fields of approximate dynamic programming [9] and reinforcement learning [10], especially for complicated systems for which a model is not known or only available through simulation. However, unlike Reinforcement Learning we are dealing with a non-interactive learning scenario, where we cannot choose how to interact with the system while samples are collected. Specifically, the learning process occurs offline and in batch. Further, since we have partial knowledge about the MDP (e.g., the immediate cost function), we use a model-based method [11] similar to the *certainty equivalence method* [12] where we estimate the most likely transition probabilities for the unknown part of the model. Unlike the dominant approach that uses function approximations to represent the value function (or Q-values) [13–15] and selects the action based on the greedy policy with respect to the estimated values, we directly represent a policy mapping states to actions. Specifically, we use supervised learning to train a policy using a dataset of "posterior optimal" actions computed according to the learned MDP model. The policy compactly represents the available information and, in our application, empirically performs better than models fitted to the Q-values. Further, in this way we can directly analyze the structure of policy being used, thus simplifying the deployment on a vehicle.

Our approach is similar to a line of research in the planning community [6, 16–18], where researchers have tried to learn strategies to solve planning problems in a fixed domain by learning from particular solutions to training problems sampled from the same domain. Specifically, our work is most closely related to [6], where they use a sample-based approach to learn a policy for a multiple-battery system modeled as an MDP. However, since we are dealing with electric vehicles we need to optimize charge/discharge cycles while they focus on the discharge aspect only. Consequently, we use a quadratic objecting function, while they optimize for the plan length. Further, their work is based on synthetic data generated from a known model, while we face the problem of learning a probabilistic model from real-world crowdsourced data, which creates additional challenges such as feature selection. Furthermore, in [6] they use as training examples sequences of state-action pairs where the actions are obtained optimizing in hindsight for a single sample realization of the randomness, i.e. the training set is generated using an optimal omniscient policy that knows the future ahead of time. Although the method is shown to perform well in practice, it doesn't provide any theoretical guarantee. As a counterexample, consider a simple MDP modeling a lottery with an expected negative return, where the actions are either bet on a number or not to play at all. Given any realization of the randomness, the optimal omniscient policy would always suggest to bet (since it knows the future), but the optimal risk-neutral strategy is not to play the lottery, and therefore it cannot be learned from such training examples. In contrast, we also use a form of hindsight optimization, but we jointly consider all the samples, using a sample-based approximation for the expectation that provably converges to the true optimal value as the number of samples grows.

In the remainder of the paper, we will describe how we apply this general approach to the battery management application.

3 Problem Description

There are two main sources of inefficiencies in batteries. The first one is that batteries have an internal resistance R_{int}, and therefore they dissipate power as heat as $R_{int}i^2$ when charged or discharged with a current i. Secondly, the capacity of a battery is related to the rate at which it is discharged by Peukert's Law. In particular, the faster a battery is discharged with respect to the nominal rate (by pulling out a higher current), the smaller the actual delivered capacity is. The effect depends on the chemical properties of the battery and is exponential in the size of the current. Therefore, substantial

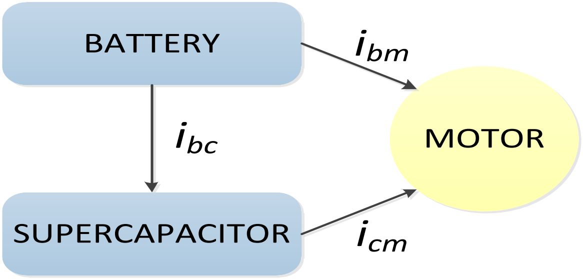


Fig. 1. Architecture of the battery system and sign convention used (a positive number indicates current flowing in the direction of the arrow)

savings can be obtained by reducing the current output from the battery used to achieve a certain desired power.

One promising direction towards improving battery efficiency are multiple-battery systems such as the ones proposed in [4] and [5], which integrate a standard battery with one or more supercapacitors, as depicted in Figure 1. Intuitively, the idea is that the battery is good at holding the charge for long times, while the supercapacitor is efficient for rapid cycles of charge and discharge. Using the supercapacitor as a buffer, high peaks in the battery's charge and discharge currents can be reduced, thus reducing the losses due to Peukert's Law and the internal resistance. In fact, supercapacitors are not affected by Peukert's Law and behave like an ideal battery. Furthermore, this can substantially increase the lifespan of batteries because of the reduced number of full charge-discharge cycles the battery must handle (supercapacitors on the other hand can handle millions of full charge/discharge cycles). Improvements in battery efficiency lead to reduced costs, increased range, and therefore more practical electric vehicles.

While the savings obtained with multi-battery systems can be substantial, they heavily depend on how the system is managed, i.e. on the strategy used to charge and discharge the capacitor. Managing such systems is non-trivial because there is a mix of vehicle acceleration and regenerative braking (when power can be stored in the battery system) over time, and there is a constraint on the maximal charge the capacitor can hold. For instance, keeping the capacitor close to full capacity would allow the system to be ready for sudden accelerations, but it might not be optimal because there might not be enough space left to hold regenerative braking energy. Intelligent management algorithms therefore need to analyze driving behavior and vehicle conditions (speed, acceleration, road conditions,..) in order to make informed decisions on how to allocate the power demand. Intuitively, the system needs to be able to predict future high-current events, preparing the supercapacitor to handle them and thus reducing the energy losses on the battery. The results can be quite impressive. In figure 2 we show the battery output over a 2 minutes window of a real world trip, when no capacitor is used, when it is managed with a naive buffer policy (charging the capacitor only from regenerative braking, and utilizing the capacitor whenever there is demand and energy available), and when it is managed by our novel system `DPDecTree`. While the total power output is the same in all 3 cases, when the system is managed by `DPDecTree` the output tends to be more constant over time, thus reducing the energy wasted due to battery inefficiencies.

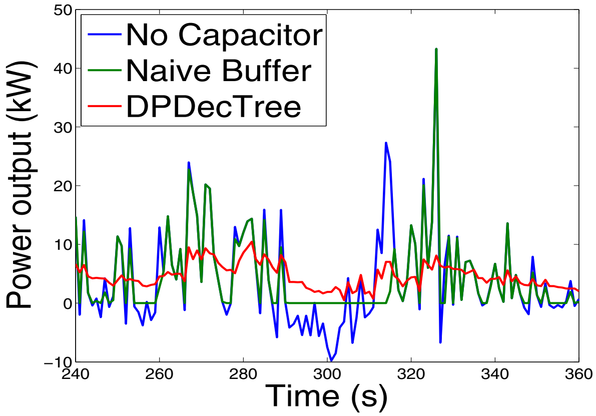


Fig. 2. Battery power output over time. A smoother more constant-like curve means reduced losses due to battery inefficiencies.

4 Modeling

To formalize the battery management problem described earlier, we consider a discrete time problem where decisions need to be taken every δ_t seconds. A decision is a choice for the variables (i_{bc}, i_{bm}, i_{cm}) in Figure 1, where i_{bc} is the current flowing from the battery to the capacitor, i_{bm} and i_{cm} are the currents from the battery and capacitor to the motor, respectively. These variables must satisfy certain constraints, namely the capacitor cannot be overcharged or overdrawn and the energy balance must be preserved. As a performance metric, we consider the i^2-score proposed in [4] and used in the Chargecar contest [8], where the objective is to minimize the sum of the squared battery output current $(i_{bc} + i_{bm})^2$ over time. Intuitively, reducing the i^2-score means reducing the energy wasted as heat and due to Peukert's Law, as well as increasing battery lifespan [19].

We first consider a simplified setting where we assume to know the future energy demand of the motor (positive when energy is required for accelerations, negative when energy is recovered from regenerative braking) ahead of time. This translates into a deterministic planning problem because we assume there is no more randomness involved. By computing the optimal sequence of actions (since the problem is deterministic, we don't need a policy), we obtain a lower bound on the i^2-score that is achievable in real-world problems where the future demand is not known.

4.1 A Quadratic Programming Formulation

Consider a single trip, where T is the number of discrete time steps (of size δ_t) in the control horizon. Let C_{max} be the maximum charge the capacitor can hold. As previously noted in [20], the problem can be formalized as a Quadratic Program. Specifically, we wish to minimize

$$\min \sum_{t=0}^{T-1} \left(i_{bc}(t) + i_{bm}(t)\right)^2$$

subject to

$$i_{cm}(t) + i_{bm}(t) = d(t),\ \forall t = 0, \cdots, T-1 \tag{1}$$

$$0 \leq \sum_{k=0}^{t} i_{bc}(k) - i_{cm}(k) \leq C_{max}/\delta_t,\ \forall t = 0, \cdots, T-1 \tag{2}$$

where $d(t)$ is the motor demand at time step t. The first set of constraints (1) requires that the demand $d(t)$ is met at every time step $t = 0, \cdots, T-1$. The second set of constraints (2) ensures that the capacitor is never overcharged or overdrawn (the capacitor is assumed to be empty at the beginning of the trip, and not to lose charge over time). Notice that the battery charge level over time is completely determined by the decision variables, and does not affect the i^2-score.

Reducing the Dimensionality. We introduce a new set of variables

$$\Delta(t) = i_{bc}(t) - i_{cm}(t), t = 0, \cdots, T-1$$

and using (1) we can rewrite the objective function as

$$\sum_{t=0}^{T-1} \left(i_{bc}(t) + i_{bm}(t)\right)^2 = \sum_{t=0}^{T-1} \left(i_{bc}(t) + d(t) - i_{cm}(t)\right)^2 = \sum_{t=0}^{T-1} (\Delta(t) + d(t))^2$$

Further, the constraints (2) can be rewritten in terms of $\Delta(t)$ as

$$0 \leq \sum_{k=0}^{t} \Delta(k) \leq C_{max}/\delta_t,\ \forall t = 0, \cdots, T-1 \tag{3}$$

In this way we have simplified the problem from $3T$ variables $\{(i_{bc}(t), i_{bm}(t), i_{cm}(t)), t = 0, \cdots, T-1\}$ to T variables $\{\Delta(t), t = 0, \cdots, T-1\}$.

The resulting Quadratic Programs can be solved to optimality using standard convex optimization packages, but long trips can take a significant amount of time (see comparison below). Since we will later consider the stochastic version of the planning problem, we consider an alternative approximate solution technique that takes into account the sequential nature of the problem and generalizes to the stochastic setting.

4.2 A Dynamic Programming Solution

A faster but approximate solution can be obtained via Dynamic Programming by discretizing the capacity levels of the supercapacitor with a step δ_C and then recursively solving the Bellman equation

$$J(t,C) = \min_{0 \le C' < N} \left\{ \left(d(t) + \delta_C/\delta_t(C'-C)\right)^2 + J(t+1,C') \right\}$$

for $t = 0, \cdots, T-1$, with boundary condition

$$J(T,C) = 0 \,\forall\, C$$

If the maximum capacity C_{max} is discretized with N steps, the complexity is $O(N^2T)$ for a trip of length T. This method is faster than solving the previous Quadratic Program directly, even though the solution is suboptimal because of the discretization. Furthermore, the DP solution does not just provide a sequence of "optimal" actions, but an actual policy that gives the action as a function of the current capacity level (for a fixed load profile).

Choosing the Discretization Step. There is a tradeoff involved in the choice of the discretization step δ_C of the capacity level. The smaller δ_C is, the better is our approximation to the original QP, but the running time also grows quadratically with $1/\delta_C$.

In order to choose the proper value of δ_C, we solved a representative subset of 54 trips (see below for the dataset description) using the QP solver in the package CVXOPT [21]. We obtained a total i^2-score of $3.070 \cdot 10^8$ in about 11 minutes. Using our DP solver with $N = 90$ steps, we obtained a score of $3.103 \cdot 10^8$ in 15 seconds; with $N = 45$ steps we obtained $3.197 \cdot 10^8$ in about 3 seconds. This experiment empirically shows that our DP based solver is about 2 orders of magnitude faster than solving the quadratic program directly, and provides solutions that are close to optimal. These experiments will guide the choice of the discretization step also for the original stochastic setting where the demand is not known ahead of time.

Robustness. Since we will later use supervised learning techniques to learn a policy from a training set, we are interested in measuring how robust is the policy to implementation errors on the actions. The sensitivity plot in Figure 3 is obtained by artificially adding i.i.d. Gaussian noise with variance σ^2 to the optimal action given by the optimal omniscient policy. The plot is averaged over a subset of trips and shows that the performance degrades smoothly as a function of the variance of the implementation error.

Rolling Horizon. In order to compute the optimal omniscient policy, we need to know the entire future demand $\{d(t), t = 0, \cdots, T-1\}$ ahead of time. Relaxing this assumption, we now assume to know the future demand only for a window of M steps. We then use a rolling horizon policy, where at every step t we replan computing the optimal sequence of actions for the next M steps, taking the first one. In Figure 4 we show how the performance improves as the length of the rolling horizon window M increases. In particular, notice that if we were able to predict (exactly) the future demand for the next 30 steps (corresponding to 30 seconds), on average we would lose less than 5% over the optimal omniscent policy (that knows the entire future demand ahead of time). This experiment suggests than even a fairly limited probabilistic model that can predict the future demand for a few seconds could provide substantial energy savings, and motivates our search for an MDP model.

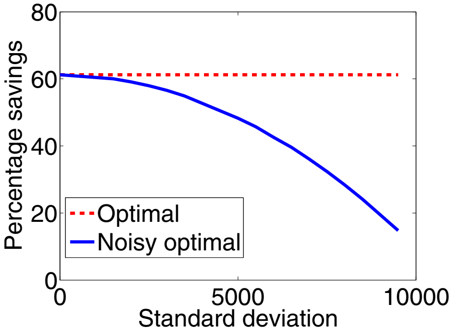


Fig. 3. Performance with noise in the implementation of the optimal policy. On the y-axis is the percentage reduction in the i^2-score with respect to baseline (no capacitor).

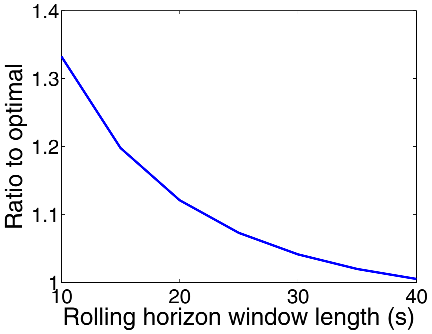


Fig. 4. Performance improvement as the length of the rolling horizon window increases

5 Probabilistic Planning

An MDP Model. In general, we cannot know in advance what will be the future demand (load profile), but we can assume the existence of an underlying stochastic model, from which the trips and driving behaviors we observe are sampled from. Specifically, we consider a *state space* $S = \mathcal{F} \times [0, C_{max}]$, where $\mathcal{F}$ is a *feature space*. The idea is to use a set of features $\mathbf{f} = (f^1, \dots, f^K) \in \mathcal{F}$ and a capacity level $0 \le c \le C_{max}$ to represent the state of the electric vehicle at any given time. The features we use are driver ID (and type of vehicle), GPS latitude and longitude, direction, speed, acceleration, altitude, instantaneous demand d, past average demand, time of the day. According to the problem definition, for any state $s = (\mathbf{f}, c) \in \mathcal{S}$, there exists a set of admissible actions $\mathcal{A}(s) = \{\Delta, -c \le \Delta \le C_{max} - c\}$, i.e. the admissible changes of the capacitor level that satisfy the constraints of the problem.

Our underlying assumption is that there exist a probabilistic model describing the evolution of the state $P\left(s_{t+1} | s_0, \cdots, s_t, \Delta_0, \cdots, \Delta_t\right)$ as a function of the previous history and the sequence of action $\Delta_0, \cdots, \Delta_k$ taken. Further, we assume that

$$\begin{aligned} &P\left(s_{t+1}|s_0,\cdots,s_t,\varDelta_0,\cdots,\varDelta_t\right) = \\ &\quad P\left(\mathbf{f}_{t+1}|\mathbf{f}_0,\cdots,\mathbf{f}_t\right)P(c_{t+1}|c_t,\varDelta_t) \end{aligned} \tag{4}$$

that is the evolution of $\mathbf{f}_t$ is independent of c_t and the actions $\varDelta_t$ taken (equivalently, we assume the driving behavior and road conditions do not depend on the capacitor charge levels). On the other hand, according to the problem description c_{t+1} depends *deterministically* on the past, specifically $c_{t+1} = c_t + \varDelta_t$. In this way, $P(c_{t+1}|c_t,\varDelta_t) = 1$ if and only if $c_{t+1} = c_t + \varDelta_t$.

In this MDP framework, a energy management system is a function mapping histories to a feasible action, i.e. a history-dependent feasible policy [22]. Upon defining an immediate cost $c(\varDelta, s, s') = (d + \varDelta)^2$ for transitioning from state s to state s' when taking action $\varDelta$ (equal to the squared current output from the battery), an optimal energy management system can be defined as one minimizing the total expected cost.

A Sample-Based Approach. Since the probabilistic model $P\left(\mathbf{f}_{t+1}|\mathbf{f_0},\cdots,\mathbf{f}_t\right)$ is unknown, we use a sample-based approach where we leverage a large dataset of commuter trips crowdsourced in the United States and available online [8] in order to learn it from the data. Specifically, we assume that each trip in the dataset corresponds to one particular realization of the underlying stochastic process, e.g. a sampled trajectory of length T_i in the feature space $\mathcal{F}$. In particular, we project the trip data on the feature space $\mathcal{F}$, generating a trajectory $\mathcal{T}_i = (\mathbf{f}_0, \mathbf{f}_1, \cdots, \mathbf{f}_{T_i-1})$ where for each time step $\mathbf{f}_t \in \mathcal{F}$. Then, for any sequence of actions $(\varDelta_0, \cdots, \varDelta_{T_i-1})$ and initial capacity level c_0, we can generate the corresponding trajectories in the *state* space $((\mathbf{f}_0, c_0), (\mathbf{f}_1, c_1), \cdots, (\mathbf{f}_{T-1}, c_{T-1}))$ according to (4).

Let $\mathcal{T}_1, \cdots, \mathcal{T}_K$ be the sample trajectories available. For any state $s = (\mathbf{f}, c) \in S$ we define the multiset

$$\mathcal{N}(\mathbf{f}) = \bigcup_{i=1}^{K} \{\mathbf{f}_{t+1} | \mathbf{f}_t \in \mathcal{T}_i, ||\mathbf{f}_t - \mathbf{f}||_\infty < \epsilon\} \subseteq \mathcal{F}$$

that when $\epsilon = 0$ corresponds to the set of feature vectors that we have observed occurring immediately after $\mathbf{f}$ in the sample trajectories. In practice, since our feature space is continuous we choose $\epsilon > 0$ to discretize the space, so that two feature vectors are considered to be the same if they are close enough (e.g., when a driver comes to an intersection with approximately the same speed, acceleration, etc.). We use k-d trees [23] to speed up the computation of $\mathcal{N}(\mathbf{f})$ for all observed feature vectors $\mathbf{f} \in \bigcup \mathcal{T}_i$, and in our experiments ϵ is set to one thousandth of the average distance between consecutive feature vectors in the available trajectories. Similarly, we can define a multiset of possible successors in the state space as

$$\mathcal{S}(s) = \mathcal{S}(\mathbf{f}, c) = \bigcup_{c'=0}^{C_{max}} \{(\mathbf{h}, c') | \mathbf{h} \in \mathcal{N}(\mathbf{f})\}$$

Posterior Optimal Actions. We can then define sample-based Dynamic Programming equations as follows

$$V(s) = \min_{\Delta \in \mathcal{A}(s)} \left(\frac{1}{|\mathcal{S}(s)|} \sum_{s' \in \mathcal{S}(s)} V(s') + c(\Delta, s, s') \right)$$

and solve for the "posterior optimal action"

$$\Delta^*(s) = \arg \min_{\Delta \in \mathcal{A}(s)} \left(\frac{1}{|\mathcal{S}(s)|} \sum_{s' \in \mathcal{S}(s)} V(s') + c(\Delta, s, s') \right)$$

This approach has the nice theoretical property that the sample-based approximation converges to the true DP equations (for the discretized MDP) in the limit of infinite samples. Similarly, $\Delta^*(s)$ converges to the optimal action as more samples are collected. In contrast, separately optimizing for the single realizations as in [6] (which corresponds to choosing $\mathcal{N}(\mathbf{f_t}) = \mathbf{f_{t+1}}$) doesn't necessarily converge to the true optimal action as $K \to \infty$, although it has been shown to work well in practice.

Regressing the Optimal Policy. Using the available sample trajectories $\mathcal{T}_1, \cdots, \mathcal{T}_K$, we generate a labeled training set[1] of (state,optimal action) pairs by solving the corresponding sample-based DP equations using value iteration (notice that the "empirical" MDP can have loops, so we cannot solve it in one pass). Specifically, we compute $\Delta^*(\mathbf{f}, c)$ for every $\mathbf{f} \in \mathcal{T}_i$ and for every capacity level $c \in [0, C_{max}]$. We then use supervised learning to learn the relationship between a state $s = (\mathbf{f}, c) = (f^1, \ldots, f^K, c) \in S$ and the corresponding optimal action Δ. Notice that the particular structure of the problem allows us to artificially generate N times more data points than what we originally started with. Experimentally, we have seen this to be a crucial improvement in order for the supervised learning algorithm to correctly understand the role of the capacity level c. In particular, we found that generating a dataset just using the optimal sequence of actions $\mathbf{a}^*$ for each trajectory $(\mathbf{f}_0, \mathbf{f}_1, \cdots, \mathbf{f}_{T-1})$ is not sufficient to achieve good performance.

The quadratic nature of the cost function gives us further insights on the performance of the supervised learning method used. In particular, the mean-squared error (MSE) is an important error metric in this case, because by reverse triangular inequality

$$||(\mathbf{a}^* + \mathbf{d}) - (\widehat{\mathbf{a}} + \mathbf{d})||_2 = ||\mathbf{a}^* - \widehat{\mathbf{a}}||_2 \geq ||\widehat{\mathbf{a}} + \mathbf{d}||_2 - ||\mathbf{a}^* + \mathbf{d}||_2$$

where $\mathbf{a}^* = (\Delta_0^*, \cdots, \Delta_{T-1}^*)$ is the optimal sequence of actions, $\widehat{\mathbf{a}}$ is the sequence of actions given by regression, and $\mathbf{d} = (d(0), \cdots, d(T-1))$ is the demand vector. This gives

$$||\widehat{\mathbf{a}} + \mathbf{d}||_2 \leq ||\mathbf{a}^* - \widehat{\mathbf{a}}||_2 + ||\mathbf{a}^* + \mathbf{d}||_2$$

so that $||\mathbf{a}^* - \widehat{\mathbf{a}}||_2$ bounds the difference in terms of i^2-score between the optimal sequence of actions $\mathbf{a}^*$ and $\widehat{\mathbf{a}}$.

[1] The training dataset will be made available online.

6 Evaluation: The Chargecar Competition

As previously mentioned, we evaluate our method on the publicly available dataset provided by Chargecar [8], a crowdsourcing project open to the public with the goal of making electric vehicles more practical and affordable. Along with the dataset, the Chargecar project provides a simulator to evaluate the performance of power management policies on the trips contained in the dataset. Furthermore, they set up an open algorithmic challenge where the goal of the contest is to design policies that optimize the energy performance of electric vehicles, as measured in terms of the i^2-score. All the parameters of the model are set as in the competition. In particular, the supercapacitor and the battery provide 50 Watt-hour and 50000 Watt-hour, respectively. Among many other factors, the energy efficiency of multi-battery schemes depends crucially on these parameters. Understanding their interplay with smart energy management policies is one of the goals of the competition, because it would allow us to design better, more efficient electric vehicles.

Dataset. The dataset [8] contains a total of 1984 trips (with an average length of 15 minutes), subdivided into 6 separate datasets according to the driver ID. Each one of these dataset is further separated into two subsets: a training and judging set. There are 168 trips in the judging set, accounting for about 8% of the total. Using the trips contained in the training set, we generate a dataset of labeled (state, optimal action) example pairs with the method explained in the previous section. The maximum capacity level is discretized into $N = 45$ discrete steps, and the time step is $\delta_t = 1s$, such that the resulting training dataset contains 75827205 examples. Since the complete training dataset generated with the previously described approach is too big to fit into memory, we divide it according to the driver ID, generating separate training sets for each driver. When these datasets are still too big, we divide them again according to the capacity level feature (selecting entries corresponding to one or more rows of the DP tables). We then learn separate models for each one of these smaller datasets, as shown in figure 5[2].

Supervised Learning. We used non-parametric exploratory models as there is little or no prior knowledge and possibly highly non-linear interactions. In particular, we use bagged decision trees, with the REPTree algorithm as implemented in the Weka package [24] as the base learner. REPTree is a fast regression tree learner that uses information gain as the splitting criterion and reduced-error pruning (with backfitting). Following standard practice, the parameters were set by 5-fold crossvalidation, selecting the model with the best MSE score. We call the resulting policy `DPDecTree`. Using decision trees, we can represent and evaluate the policy efficiently in order to compute the optimal action. In contrast, it can be impractical to compute an action solving an optimization problem based on a estimated future demand, because in a real-world setting it might not be feasible to solve such problems with a high frequency on a car.

Evaluation. We evaluate the performance of `DPDecTree` on the separate judging set of trips using the simulator. If the action suggested by `DPDecTree` would overcharge

[2] A separate default model is trained for previously unseen driver IDs.

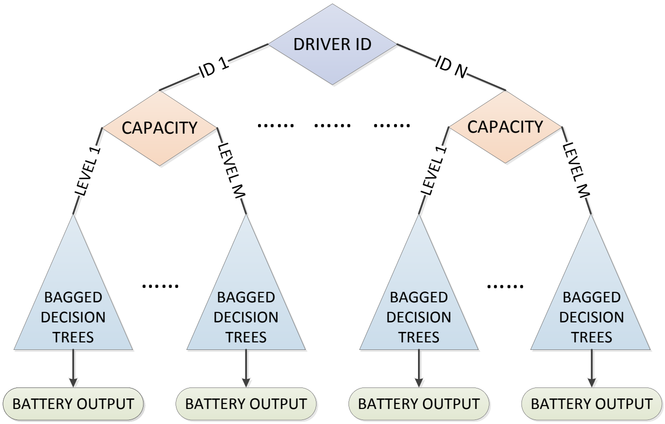


Fig. 5. An overview of our intelligent energy management system

the capacitor, we charge it to full capacity; if it overdraws, we discharge it completely. However, this situation is rare and DPDecTree gives feasible actions 99.8% of the time when evaluated on the judging set. We compare our solution against MPL, the current winning algorithm in the competition at the moment of this paper submission, and to a simple buffer policy. The MPL policy is based on a large table of thresholds for the battery output $i_{bc} + i_{bm}$, chosen according to driver ID, speed, demand and GPS coordinates. The naive buffer policy charges the capacitor only from regenerative braking (i.e, when the demand is negative) and when there is an energy demand, it utilizes the capacitor first. We also provide the i^2-score when the supercapacitor is not used (or is not available) as a baseline.

As can be seen from table 1, DPDecTree leads to significant energy savings with respect to the simple buffer policy. Although is it often far from the upper bound represented by the optimal Omniscient policy (except on the *mike* dataset where they differ only by 5%), the good performance of DPDecTree suggests that it is very effective at predicting future energy demands, and that the policy learned from the training examples generalizes to new, previously unseen scenarios. Further, DPDecTree improves over MPL in 5 out of 6 datasets (each one corresponding to a different driver). We believe the problem with the *arnold* dataset is the presence of different driving behaviors in the same dataset that the learning algorithm was not able to separate using the available covariates.

In the bottom row of Table 1 we provide the resulting scores for the entire judging dataset. On average, DPDecTree leads to a 2.5% improvement on the i^2-score with respect to MPL. According to a one-sided paired t-test, the difference is statistically significant (with P-value 0.0058). The result is significative also according to a Wilcoxon signed-rank test (with P-value 0.00028) [25]. While there is still a significant 20% gap with respect to the optimal omniscent policy, it is not clear how much can still be achieved in the real online setting where the future demand is not known ahead of time. These results suggest that there is a great potential for including

Table 1. Results on driver specific judging datasets. i^2-scores are in $10^8 \cdot A^2s$.

Dataset	DPDecTree	MPL	Naive Buffer	Baseline	Omniscient
alik	**4.233**	4.435	7.533	8.424	3.196
arnold	4.090	**3.946**	8.402	8.894	3.332
mike	**3.245**	3.290	4.874	5.128	3.083
thor	**1.648**	1.787	3.931	4.596	1.413
illah	**0.333**	0.353	0.751	0.856	0.211
gary	**2.000**	2.146	5.187	5.857	1.261
Total:	**15.549**	15.957	30.678	33.755	12.496

routing information (e.g., from a car navigation system) into the problem, since they would bring the policies closer to the omniscient case. However, in this paper we assume that routing information is not available, as in the competition. It would also be very interesting to explore the possibility of improving the policy by learning from new data as it becomes available during the policy evaluation phase, possibly with an incremental learning approach.

7 Conclusions

In this paper we have presented an effective solution to the problem of managing multi-battery systems in electric vehicles. Our novel intelligent energy management system is evaluated on a large dataset of commuter trips crowdsourced in the United States. Our approach is completely data-driven and can be expected to improve as more data is being collected and becomes available.

Our method combines several existing approaches to solve a problem that we model as an MDP with unknown transition probabilities. We use a sample-based approach, where samples are not generated from an analytic model or from a simulator but given as part of a dataset. By observing the empirical transition probabilities of a discretized problem, we solve sample-based dynamic programming equations using value iteration. Thanks to the special structure of the problem and its indipendent dynamics assumption, we can generate more artificial data points from the samples by exploiting the information contained in the dynamic programming tables. The optimal posterior actions given the observed samples are then combined to form a policy for the original problem. In order to do this, we use supervised machine learning techniques to build a regression model that gives the action as a function of the state of the system. The obtained policy is evaluated on a separate set of real world trips, where it is shown to generalize to situations that were previously unseen in the training set. Our novel system is shown to outperform the leading algorithms that were previously proposed as part of an open algorithmic challenge.

Acknowledgements. This work was supported by NSF Expeditions in Computing Grant 0832782.

References

1. Gomes, C.: Computational Sustainability Computational Methods for a Sustainable Environment,Economy, and Society. The Bridge, National Academy of Engineering 39(4) (2009)
2. Sachenbacher, M., Leucker, M., Artmeier, A., Haselmayr, J.: Efficient energy-optimal routing for electric vehicles. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
3. Apple, J., Chang, P., Clauson, A., Dixon, H., Fakhoury, H., Ginsberg, M., Keenan, E., Leighton, A., Scavezze, K., Smith, B.: Green driver: Ai in a microcosm. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
4. Dille, P., Duescher, M., Nourbakhsh, I., Podnar, G., Schapiro, J.: Evaluating the urban electric vehicle. Carnegie Mellon Technical Report (February 2010)
5. Kötz, R., Müller, S., Bärtschi, M., Schnyder, B., Dietrich, P., Büchi, F.N., Tsukada, A., Scherer, G.G., Rodatz, P., O., Garcia, o.: Supercapacitors for peak-power demand in fuel-cell-driven cars. In: ECS Electro-Chemical Society, 52nd Meeting, San Francisco (2001)
6. Fox, M., Long, D., Magazzeni, D.: Automatic construction of efficient multiple battery usage policies. In: Proc. Int. Conf. on Aut. Planning and Scheduling, ICAPS (2011)
7. Jongerden, M., Haverkort, B., Bohnenkamp, H., Katoen, J.P.: Maximizing system lifetime by battery scheduling. In: IEEE/IFIP International Conference on Dependable Systems And Networks, DSN 2009, pp. 63–72. IEEE (2009)
8. CreateLab. The chargecar project (January 2012), http://www.chargecar.org
9. Powell, W.B.: Approximate Dynamic Programming: Solving the curses of dimensionality, vol. 703. Wiley-Blackwell (2007)
10. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, vol. 28. Cambridge University Press (1998)
11. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
12. Kumar, P.R., Varaiya, P.: Stochastic systems: estimation, identification and adaptive control. Prentice-Hall, Inc., Upper Saddle River (1986)
13. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic programming (1996)
14. Tsitsiklis, J.N., Van Roy, B.: An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control 42(5), 674–690 (1997)
15. Gordon, G.J.: Stable function approximation in dynamic programming. In: Machine learning: proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9-12, p. 261. Morgan Kaufmann (1995)
16. Khardon, R.: Learning to take actions. Machine Learning 35(1), 57–90 (1999)
17. Khardon, R.: Learning action strategies for planning domains. Artificial Intelligence 113(1), 125–148 (1999)
18. Yoon, S.W., Fern, A., Givan, R.: Using learned policies in heuristic-search planning. In: Proceedings of the 20th IJCAI (2007)
19. Peterson, S.B., Apt, J., Whitacre, J.F.: Lithium-ion battery cell degradation resulting from realistic vehicle and vehicle-to-grid utilization. Journal of Power Sources 195(8), 2385–2392 (2010)
20. Daniel Wong. Energy management optimization in electric vehicles using model predictive control. Unpublished technical report (2011)
21. Dahl, J., Vandenberghe, L.: Cvxopt: A python package for convex optimization. In: Proc. Eur. Conf. Op. Res (2006)

22. Puterman, M.L.: Markov decision processes: Discrete stochastic dynamic programming. John Wiley & Sons, Inc. (1994)
23. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM 18(9), 509–517 (1975)
24. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
25. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83 (1945)

Policy Iteration Based on a Learned Transition Model

Vivek Ramavajjala and Charles Elkan

Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92092

Abstract. This paper investigates a reinforcement learning method that combines learning a model of the environment with least-squares policy iteration (LSPI). The LSPI algorithm learns a linear approximation of the optimal state-action value function; the idea studied here is to let this value function depend on a learned estimate of the expected next state instead of directly on the current state and action. This approach makes it easier to define useful basis functions, and hence to learn a useful linear approximation of the value function. Experiments show that the new algorithm, called NSPI for next-state policy iteration, performs well on two standard benchmarks, the well-known mountain car and inverted pendulum swing-up tasks. More importantly, the NSPI algorithm performs well, and better than a specialized recent method, on a resource management task known as the day-ahead wind commitment problem. This latter task has action and state spaces that are high-dimensional and continuous.

1 Introduction

> "...the state of a system is often summarized by certain features or basis functions that capture the state's salient properties. The selection of suitable features is often the *condicio sine qua non* for practical success. The choice of features is almost always influenced by sound engineering understanding of the problem domain, but automating this process would be a major step ahead." (Tsitsiklis, 2010) [17].

This paper takes a step forward towards the goal of automating the process of choosing useful basis functions in reinforcement learning.

The objective of a reinforcement learning algorithm is to acquire a policy for choosing actions that can control an agent to desired goal states. A value function is a function that estimates the long-term reward, as opposed to the immediate reward, of a given state, or of a given state combined with a given action. A Q function is a value function that maps each state-action pair to a real number that measures the long-term utility of taking that action in that state. Approaches that learn value functions approximately, and in particular methods that learn Q functions approximately, have been used to solve reinforcement learning problems successfully. While other functional forms have also been used, linear approximations of Q functions are popular because of convergence

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 211–226, 2012.

guarantees, ease of implementation, and low computational complexity [8]. A linear approximator represents the value of a state-action pair as a weighted sum

$$Q(s,a) = \sum_{j=1}^{m} \phi_j(s,a)v_j = \phi(s,a) \cdot v$$

of m predetermined basis functions where v is a real-valued vector of length m. Several approaches to learn v, notably approximate policy iteration [8], have been investigated intensively but the choice of the basis functions themselves has been studied less. In this paper we suggest a process to define useful basis functions, and we present empirical results that demonstrate the effectiveness of the approach.

The rest of this paper first reviews Markov decision processes (Section 2), and then describes the proposed process for defining useful basis functions (Section 3). The specific algorithm that we suggest, called NSPI, is presented in Section 4, which also discusses related research briefly. Then Section 5 presents experimental results on standard benchmarks, and Section 6 shows the effectiveness of the approach on a high-dimensional application.

2 Markov Decision Processes

The context of our work is reinforcement learning (RL) from historical data, which is often called batch RL. Although the majority of research in recent decades has concerned RL with interactive exploration of the environment, non-interactive RL tasks were in fact the applications that motivated the original invention of Markov decision processes (MDPs) [6].

An MDP is the formalization of the scenario underlying an RL task. Precisely, an MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition model with $\mathcal{P}(s,a,s')$ being the probability of making a transition from s to s' on taking action a, $\mathcal{R}$ is the reward model with $\mathcal{R}(s,a,s')$ being the immediate reward associated with transitioning from s to s' on taking action a, and $\gamma \leq 1$ is the discount factor. We assume that the MDP has an infinite horizon and that future rewards are discounted exponentially with the discount factor γ. For problems with a goal or sink state, the discount factor may be equal to 1, while in problems without any goal or sink state, the discount factor must be strictly less than one. The expected reward for taking an action a in a state s is

$$R(s,a) = \int_{s'} \mathcal{R}(s,a,s')\mathcal{P}(s,a,s')ds'.$$

A stationary policy is a mapping $\pi : \mathcal{S} \to \Omega(\mathcal{A})$ where $\Omega(\mathcal{A})$ is the space of probability distributions over the action space; $\pi(a;s)$ is the probability of choosing action a in state s. A deterministic stationary policy is a mapping $\pi : \mathcal{S} \to \mathcal{A}$, with $\pi(s)$ being the action the agent takes in state s. For a stationary deterministic policy π, the Q function $Q^{\pi}(s,a)$ that represents the utility of taking action a in state s can be expressed as

$$Q^{\pi}(s,a) = R(s,a) + \gamma \int_{s'} Q^{\pi}(s', \pi(s'))\mathcal{P}(s,a,s')ds'.$$

Given any Q function $Q(s,a)$, the corresponding policy is $\pi(s) = \mathrm{argmax}_a\, Q(s,a)$.

All problems considered in this paper are Markov decision processes where the state, action, reward and next state can be observed in every transition. The state and action spaces $\mathcal{S}$ and $\mathcal{A}$ are real-valued and may be high-dimensional. The transition model $\mathcal{P}$ and the reward model $\mathcal{R}$ are assumed to be unknown.

3 Defining Useful Basis Functions

An optimal policy, and hence an optimal Q function, specifies an action that moves the agent from the current state to the best possible next state (which may be the same as the current state). The long-term value of a state in an MDP does not depend on the state from which it was reached, or on the action taken to reach it. Thus, the next state (which is the net effect of the action taken in the current state) may be more informative than the current state and the action taken, as the argument of a Q function. Given that an approximate Q function is a weighted combination of basis functions, this observation leads to our primary intuition for defining useful basis functions: these should take as sole input an estimate of the next state that results from the action taken in the current state.

Concretely, the proposal is to write

$$Q(s,a) = Q(\hat{s}) = \sum_{j=1}^{m} \phi_j(\hat{s}) v_j$$

where $\hat{s} = f(s,a)$ for some function f is an estimate of the next state reached from the current state s by taking action a. In general, the next state is not determined deterministically by s and a, so it is more precise to say that $\hat{s}$ is an estimate of the expected next state. In this paper we assume that states are real-valued vectors, so expectations are well-defined, subject to some minor technical conditions. The transition model that describes the next state is the component $\mathcal{P}$ of the Markov decision process. The essence of reinforcement learning is that $\mathcal{P}$ is not known, but an approximation of it can be estimated from sampled data.

Having the next state as input leads to the secondary intuition for defining useful basis functions: these should measure aspects of the next state that are correlated with its long-term value. This intuition is consistent with most previous work, so arguably it is not novel. However, it seems easier to put into practice when it is applied to the estimated expected next state, rather than to the current state and action. The idea can be made more precise by considering two different varieties of task (again this distinction is not novel). The first variety consists of tasks where there are specific terminal states. For these tasks, each basis function should describe a small neighborhood of states. Then, the trained coefficient of each basis function can indicate the proximity of the neighborhood to good and/or bad ending states. The second variety consists of tasks where there are no defined terminal states, but each state generates a varying reward. For these tasks, each basis function should describe a relatively separable aspect of the following state. Then, the trained coefficients of the basis functions can collectively indicate the goodness of the expected next state. As a special case, basis functions can be components of the vector $\hat{s}$ that represents the expected next state, plus other functions that can be computed from this vector.

The first variety of tasks includes the well-known mountain car and inverted pendulum benchmarks, where a small region of the state space is the goal region. These two tasks are investigated experimentally in Section 5 below, using the primary and secondary intuitions just described. The second variety of tasks includes resource management scenarios, where stocks and flows must be controlled in order to maximize profits and minimize costs [13]. In these scenarios, aspects of the estimated state can be the levels of various resources, their prices, environmental conditions such as weather, and so on. A task of this nature is solved in detail in Section 6.

4 The NSPI Algorithm

This section describes a concrete algorithm that puts into practice the ideas of the previous section. As mentioned, the primary idea is that a Q function should be represented as $Q(s, a) = Q(\hat{s})$ where $\hat{s}$ is an estimate of the expectation of the next state. In our approach, $\hat{s}$ is a linear function of s and a. This function is used only to predict the immediate next state and not an extended path, so typically it only needs to predict transitions to states in a small neighborhood of the current state. Even domains exhibiting a high degree of nonlinearity, such as the inverted pendulum domain described below, have local transitions that are approximately linear functions of a representation of the current state and action. Of course, local linearity does not imply global linearity, and in some domains the true transition function is discontinuous.

For a batch reinforcement learning task, the training data consist of n quadruples of the form (s_i, a_i, r_i, s'_i) where s_i is a state, a_i is the action taken in that state, and r_i and s'_i are the reward and next state that were observed to ensue. The linear estimated transition model is simply the matrix T that is the least squares solution of the overdetermined system of n linear equations each one of the form

$$[s_i \; a_i]T = s'_i.$$

Then, for any state s and action a, the approximate Q function is

$$Q(s, a) = Q(\hat{s}) = \sum_{j=1}^{m} \phi_j([s \; a]T)v_j$$

where the weights v_j are learned in a second stage, separately from learning T.

When appropriate, the linear transition model can depend on a nonlinear representation of the state. For example, if one component of the state is a measured angle, then having the sine and cosine of the angle in the state representation can give additional information. If the re-representations of the state and action spaces are denoted by the functions f_s and f_a respectively, then the transition matrix T is the least squares solution of the n linear equations

$$[f_s(s_i) \; f_a(a_i)]T = s'_i$$

Algorithm 1. NSPI (next-state policy iteration)

```
// Input: training samples $D = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^n$
// $\phi$: basis functions $\phi_1$ to $\phi_m$
// $\gamma$: discount factor
// $\epsilon$: stopping criterion
// Output: weight vector $v'$ representing the learned policy

Stage 1: Solve $[f_s(s_i)\ f_a(a_i)]T = s'_i|_{i=1}^n$ for $T$
Stage 2: // LSPI
$v' \leftarrow 0$
repeat
    $v \leftarrow v'$
    $A \leftarrow 0$                         // $m \times m$ matrix
    $b \leftarrow 0$                         // $m \times 1$ column vector
    for each $(s, a, r, s') \in D$ do
        $s_{next} = [f_s(s)\ f_a(a)]T$              // estimate the next state for $s$
        $a^* = \text{argmax}_{a'}\ \phi([f_s(s')\ f_a(a')]T) \cdot v$
        $s'_{next} = [f_s(s')\ f_a(a^*)]T$            // estimate the next state for $s'$
        $A \leftarrow A + \phi(s_{next})\Big(\phi(s_{next}) - \gamma\phi(s'_{next})\Big)^T$
        $b \leftarrow b + \phi(s_{next})r$
    end for
    Solve $Av' = b$ for $v'$
until $||v - v'|| < \epsilon$
```

and the Q function for a given state s and action a is $Q(s, a) = \phi(\hat{s}) \cdot v$ where $\hat{s} = [f_s(s)\ f_a(a)]T$.

After a linear transition model T is trained, the second stage of our approach is to learn a Q function using the next-state basis function representation $\phi(\hat{s})$. This paper uses the least squares policy iteration (LSPI) method for the second stage [8]. LSPI has the advantage of not having to select parameters such as the learning rate or the number of steps to iterate over. The full proposed algorithm using next-state basis functions is shown in Algorithm 1.

Regularization can be used for learning both the transition matrix T and the weights v. For the experiments described below, regularization is not needed for learning the transition matrix, because the number n of training samples is much larger than the number of features, which is the length of the concatenated vectors $[f_s(s)\ f_a(a)]$. For learning the weights v, regularization is used by adding a scaled identity matrix λI to the matrix A, where λ is a small positive value. Empirically, regularization in learning v helps stabilize the estimates of the weights, and reduces the number of steps required to reach a stable estimate.

A practical concern with Q functions is the argmax operation used to find the optimal action. For small discrete action spaces, as in Section 5, finding an optimal action is easy, but searching in a continuous action space requires an optimization algorithm of some sort. This can be computationally expensive, especially if the space of feasible actions is constrained. Choosing the optimal action given s can be formulated as

$$a^* = \underset{a}{\operatorname{argmax}} \sum_{j=1}^{m} \phi_j([f_s(s)\ f_a(a)]T)v_j \text{ such that } x \leq Ca \leq y$$

where the matrix C and the lower and upper bound vectors x and y specify constraints on allowed action vectors. If the basis functions ϕ_j and the action representation function f_a are linear, then linear programming can find the optimal a quickly even with constraints. Otherwise, an alternative optimization approach such as an interior point method can be used, as is the case in Section 6 below.

Algorithm 1 is novel as far as we know, but of course it is not unprecedented. The fundamental aspect that makes reinforcement learning be a process of learning from data, and different from solving the MDP directly, is that the transition model $\mathcal{P}$ is unknown. RL methods are called model-based if they learn $\mathcal{P}$ explicitly in some way, and model-free if they do not [12]. (Some model-based algorithms learn to predict the immediate reward as well as the next state.) Standard least-squares policy iteration is a model-free method, while NSPI is model-based. Given a limited quantity of training samples $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^{n}$, learning both a model and a value function can maximize the amount of useful information extracted from the training data. One reason is that in many domains the transition model is close to linear, even though the optimal value function is not. Another reason is that in many domains the long-term value is a simpler function of the following state than it is directly of the current state s and an action a.

There are several RL methods that are similar in some ways to Algorithm 1. The earliest method to learn a Q function and a transition model at the same time may have been the Dyna-Q method [15]. A more recent paper that combines model-based and model-free learning is [7]. Its algorithm called AMBI is interactive, which is likely a major reason why it needs about 100 episodes to learn consistently a good policy for the mountain car task, compared to about 40 episodes for the NSPI algorithm. Other related work is [9], whose algorithm has a first stage that uses information collected about the state space to define so-called proto-value functions as basis functions, and a second stage that uses LSPI to determine a good policy based on these. Another line of research whose aim is similar to the aim here investigates methods to adapt the parameters of basis functions of a known form [11].

5 Benchmark Results

This section describes experiments applying the proposed NSPI algorithm to the mountain car problem and to the inverted pendulum swing-up problem, which are two standard benchmarks for reinforcement learning. The following section then describes applying the method to a large-scale inventory management problem that concerns managing a wind energy farm to maximize profits from sale of electricity. The first two problems require the policy to control the agent to a goal state, while the last requires the agent to maximize long-term reward. For each task, we briefly describe the problem domain, define the basis functions, explain procedures for training and testing policies, and compare the performance of learned policies with existing results.

5.1 Mountain Car

The mountain car task requires controlling a car up a hill when its own power is insufficient to climb the hill. The car must back up on an opposite hill to gain momentum, and use that momentum to accelerate to the goal position [16]. The state is a tuple (x, v) where $x \in [-1.2, 0.6]$ is the position of the car and $v \in [-0.07, 0.07]$ is the velocity of the car. The goal states are those with $x \geq 0.5$. There is an inelastic wall at the left end of the domain, i.e., if the car reaches $x = -1.2$, it is stopped and given a velocity of 0. In each state, the action is either to accelerate forward, cruise, or accelerate backwards, i.e., $a \in \{-0.001, 0, 0.001\}$. The effect of gravity at a position x is given by $-0.0025 \cos(3x)$. After each transition, the position and velocity are bounded to stay within the specified domain. Thus, the transition from a state (x_t, v_t) to the next state for a given action a is

$$\begin{aligned} x_{t+1} &= \max\{\min\{x_t + v_t, 0.6\}, -1.2\} \\ v_{t+1} &= \max\{\min\{v_t + a - 0.0025 \cos(3x_t), 0.07\}, -0.07\}. \end{aligned}$$

The reward at each step is $r_t = 0$ if $x_t \geq 0.5$ and $r_t = -1$ otherwise. Given n training samples $\{s_i, a_i, r_i, s'_i\}_{i=1}^n$, the transition matrix T is the least squares solution of the overdetermined linear system of n equations

$$[s_i \; a_i]T = s'_i.$$

Directly following the intuition suggested in Section 3 above, the basis functions are 25 Gaussian radial functions whose centers are distributed evenly over the state space. Approximately, the learned weight for each basis function indicates how good that neighborhood of the state space is. Each Gaussian has a fixed width proportionate to the distance between two neighboring centers. The basis functions are thus

$$\phi_m(\hat{s}) = \exp\left(-\frac{1}{2}(\hat{s} - c_m)\Sigma^{-1}(\hat{s} - c_m)^T\right)$$

with $c_m \in \{-1.2, -1, -0.8, \ldots, 0.6\} \times \{-0.07, -0.054, \ldots, 0.054, 0.07\}$ and

$$\Sigma = \begin{bmatrix} 0.036 & 0 \\ 0 & 0.0006 \end{bmatrix}.$$

Samples for training were collected using random trials, i.e., starting the car with a randomly chosen position and velocity and following a policy that selects actions at random. The training trials were restricted to at most 60 steps. For testing, each policy was given 100 randomly chosen starting positions and each test trial was followed for at most 500 steps. A policy that controlled the car to the goal state for all 100 test trials was considered to be successful. This experiment was repeated 10 times for each training set size. The performance for each training set size is reported as the average over the 10 experiments. All policies were learned using $\gamma = 0.95$ with strength of regularization $\lambda = 0.4$. The strength of regularization was chosen empirically to ensure that the LSPI stage of the NSPI algorithm arrives at a stable estimate of the weights.

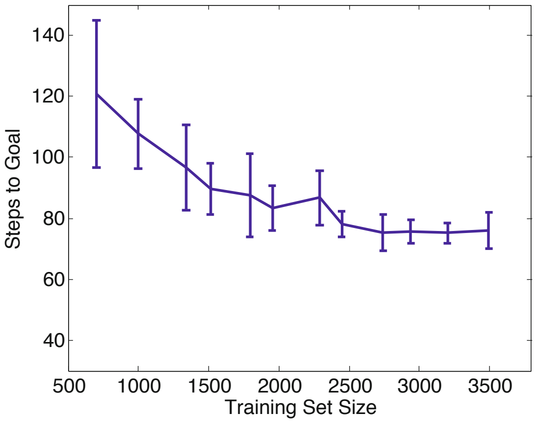


Fig. 1. Results for the mountain car benchmark: dependence of steps-to-goal on the training set size

The variation of the performance with the training set size is shown in Fig. 1. The first successful policy was learned with a training set having only 300 samples (roughly 5 random trials), though the policy was not optimal. With training set sizes of 2400 samples or more, the NSPI algorithm consistently learns a policy that controls the car to the goal state in at most 76 steps averaged across all test starting states, with a standard deviation of 3 to 6 steps. The best learned policy controls the car to the goal in an average of 68 steps, learned on a training set of 2200 samples. Some of the average steps-to-goal reported previously are 104 [10], 70 to 80 [14], and 63 [18]. In light of these comparisons, we can conclude that a close-to-optimal policy is learned consistently with about 40 training trials.

5.2 Inverted Pendulum

The inverted pendulum task involves a pendulum attached to a cart, where the agent can only apply horizontal forces to the cart. The goal is to swing the pendulum from its stable equilibrium position (the pendulum pointing downwards) to the unstable equilibrium position where the pendulum points upwards. A state is a tuple $(\theta, \dot{\theta})$ where θ is the angle measured from the unstable equilibrium position and $\dot{\theta}$ is the angular velocity. Both dimensions are unbounded. The available actions are a positive force $+10N$, a negative force $-10N$, or no force to be applied to the cart.

The dynamics for the inverted pendulum problem are given by [19] as

$$\ddot{\theta} = \frac{g\sin(\theta) - \alpha ml(\dot{\theta})^2 - \alpha\cos(\theta)u}{4l/3 - \alpha ml\cos^2(\theta)}$$

where the action $u \in \{-10, 0, 10\}$, the constant $\alpha = 1/(M + m)$, and the one-step transition function is $\theta \leftarrow \theta + \tau\dot{\theta}$ and $\dot{\theta} \leftarrow \dot{\theta} + \tau\ddot{\theta}$. Here, $M = 1.0, m = 0.1, l = 0.5$ are the masses of the cart and the pole and the length of the pole, while $\tau = 0.02$ is the time step duration, and g is the gravitational constant. The agent receives a penalty

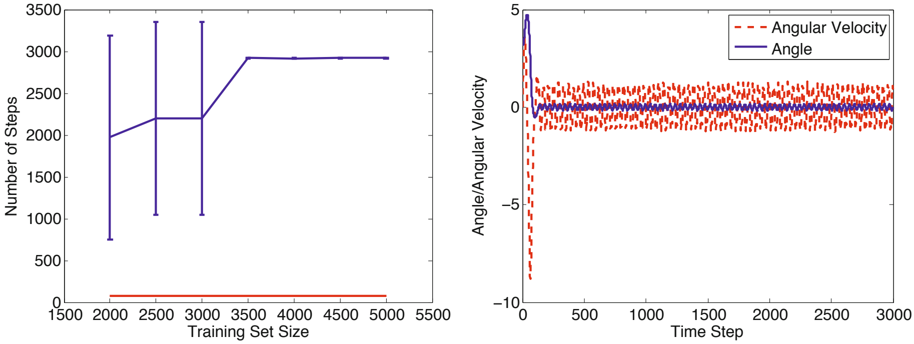


Fig. 2. Results for the inverted pendulum. In the left panel, the red line shows the number of steps needed to reach the region $(-\pi/12, \pi/12)$, and the blue line shows the number of steps for which the pendulum stayed in the region $(-\pi/12, \pi/12)$. In the right panel, the blue line shows the angle of the pendulum measured from upright, starting at π and stabilizing around 0, while the red line indicates the angular velocity.

of -1 for each step that the pendulum is below the horizontal line, and a reward of 0 for each step the pendulum is above the horizontal line, i.e. $r(\theta, \dot{\theta}) = 0$ if $\cos(\theta) > 0$; $r(\theta, \dot{\theta}) = -1$ otherwise. An implementation of the above system dynamics is available online [5].

Given n training samples $\{s_i, a_i, r_i, s'_i\}_{i=1}^{n}$, the transition matrix T is the least squares solution of the linear system of n equations

$$[f_s(s_i)\ f_a(a_i)]T = s'_i$$

where

$$\begin{aligned} f_s(s) = [&\theta, \dot{\theta}, \sin(\theta), \cos(\theta), \sin(\theta)^2, \cos(\theta)^2, \sin(\theta)\cos(\theta), \\ &\dot{\theta}\sin(\theta), \dot{\theta}\cos(\theta), \dot{\theta}\sin(\theta)^2, \dot{\theta}\cos(\theta)^2, \dot{\theta}\sin(\theta)\cos(\theta), \\ &\dot{\theta}^2\sin(\theta), \dot{\theta}^2\cos(\theta), \dot{\theta}^2\sin(\theta)^2, \dot{\theta}^2\cos(\theta)^2, \dot{\theta}^2\sin(\theta)\cos(\theta)] \\ f_a(a) = [&a\cos(\theta), a\sin(\theta)]. \end{aligned}$$

The function f_s is a nonlinear representation that maps the state to a higher-dimensional space. Though it appears complex, the mapping f_s is easy to define: it consists of the terms of the polynomial $(1 + \sin(\theta) + \cos(\theta))^2(1 + \dot{\theta} + \dot{\theta}^2)$ without the constant term. Representing the state using such a mapping captures the dependence of the transition model on higher order terms of $\dot{\theta}, \sin(\theta)$ and $\cos(\theta)$. Adopting this mapping does not require prior knowledge of the real transition model. Similarly, the function f_a maps the action to a higher-dimensional space, simply separating the action into the component directions along and perpendicular to the pendulum.

Again following the intuition suggested in Section 3 above, the basis functions consist of 25 Gaussians evenly distributed in $\{-\pi, 0, \pi\} \times \{-3, 0, 3\}$. Each Gaussian has width proportional to the distance between two neighboring centers. Note that although the Gaussians are centered inside the grid bounded by $\{-\pi, \pi\} \times \{-3, 3\}$, the values of angle and angular velocity may go outside this grid. The basis functions are thus

$$\phi_m(\hat{s}) = \exp\left(-\frac{1}{2}(\hat{s} - c_m)\Sigma^{-1}(\hat{s} - c_m)^T\right)$$

where

$$\begin{aligned} \hat{s} &= [f_s(s)\ f_a(a)]T \\ \Sigma &= \begin{bmatrix} \pi/5 & 0 \\ 0 & 3/5 \end{bmatrix} \\ c_m &\in \{-\pi, \pi/2, 0, \pi/2, \pi\} \times \{-3, -1.5, 0, 1.5, 3\} \end{aligned}$$

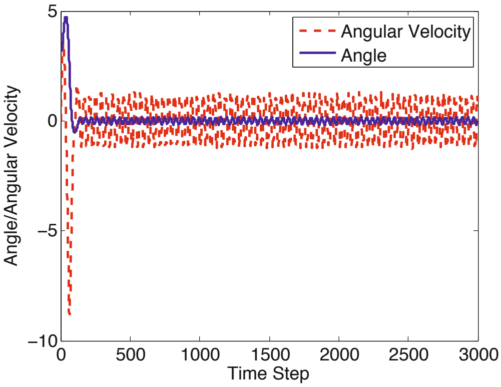
and the approximate Q function is

$$Q(\hat{s}) = \sum_{m=1}^{25} \phi_m(\hat{s}) v_m.$$

Samples for training were collected by randomly sampling the state-action space. Each sample (s, a, r, s') was generated by randomly selecting a state from $[-\pi, \pi] \times [-3, 3]$ and an action from {-10, 0, 10}. The next state and the reward were calculated following the system dynamics. For testing, each test trial began with the pendulum at the lowest position and the policy was followed for at most 3000 steps. The performance metrics tracked were the time taken to reach the goal region $(-\pi/12, \pi/12)$, and the time the pendulum stayed in this region. The experiment was repeated 10 times for each training set size. The performance for each training set size is reported as the average over the 10 experiments. All policies were learned using $\gamma = 0.95$ with strength of regularization $\lambda = 1$. The strength of regularization was chosen empirically to ensure the algorithm arrives at a stable estimate of the weights.

The variation of performance with training set size is shown in Fig. 2. A policy that controls the pendulum to the goal is learned with as few as 1500 samples. With a training set size of 3500 or more samples, the learned policies consistently control the pendulum to the goal and keep it in the upright position for an indefinite duration.

As a further test, a training dataset of 4000 samples was generated as described earlier. The policy learned from this dataset was allowed to choose actions from the set {-10, -8, -2, ..., 0, 2, ..., 10} during testing. Though the policy was trained on only the actions {-10, 0, 10}, it was still able to balance the pendulum using the new set of actions (Fig. 3). Further, it chose the new actions for 1,279 steps out of a 3000 step trial, preferring the lower magnitude actions when the pendulum was closer to the goal state. This indicates that the learned Q function correctly estimates the greater utility of lower magnitude actions closer to the goal, even though those actions were absent during training. This is because the basis functions depend only on the predicted next state, and prediction of next states is sufficiently accurate even for the new actions. For comparison, in the framework used by Lagoudakis and Parr [8] and by other researchers for the inverted pendulum task, each of the three actions {-10, 0, 10} has its own set of basis functions. Because of the explicit dependence on discrete actions as arguments, each new action requires additional basis functions in that framework.



(a)

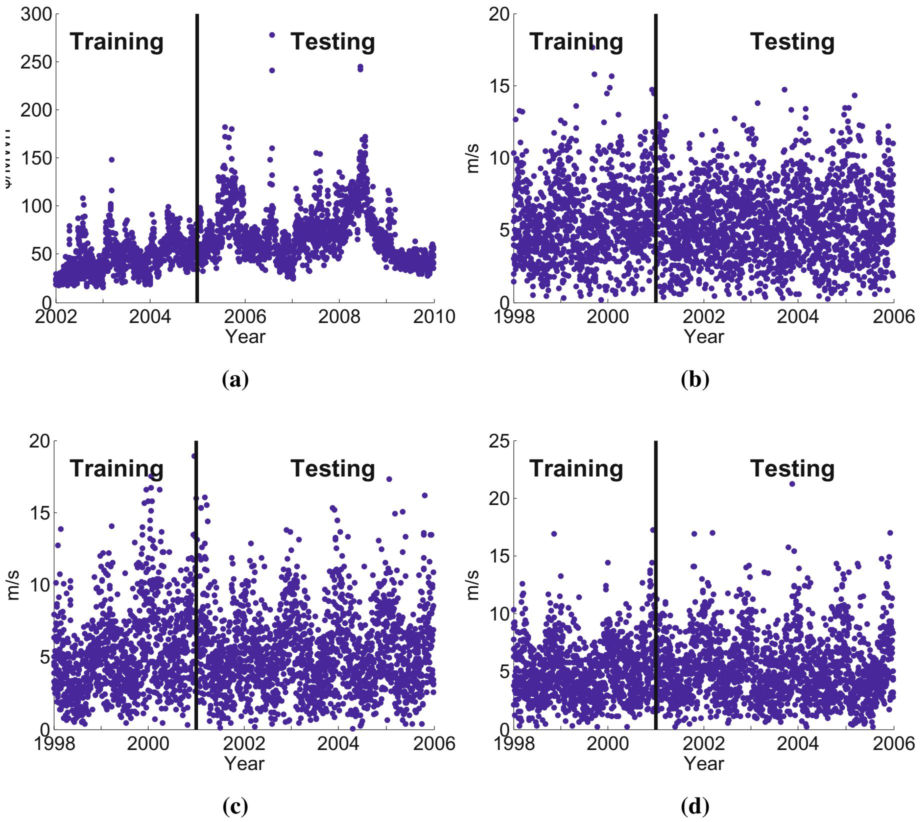
Fig. 3. Pendulum trajectory using the extended set of actions: the pendulum is still balanced, despite the policy being learned for a different set of actions

6 Management of a Wind Farm

The day-ahead wind commitment problem is a multistage stochastic optimization problem described recently [4]. The agent is a wind energy producer with a certain storage capacity for electricity. Energy markets are based around bids for future production, so producers must commit a certain amount of electricity 24 hours ahead in the day-ahead market. At the end of each day, the agent knows the hourly wind speed for the day and the hourly prices per unit of electricity for the next day.

The agent must commit to providing a certain amount of electricity for each hour of the next day. For each unit of electricity committed, it receives revenue equal to the price per unit of electricity at that hour. If the agent is unable to provide the amount committed, it must make up the difference by buying from the spot market at twice the per unit price at that hour. If it generates more electricity than promised, it may store the excess generated electricity. Storage is free but limited. If the storage capacity has been reached, then the excess electricity that cannot be stored must be dumped at the cost of $5 per unit dumped. Units are megawatt hours (MWh) and a typical price per unit is $50, so dumping is not free but not highly expensive either. The goal of the agent is to commit energy each hour of the next day in a way that maximizes profit. Note that the scenario assumes that operating costs are fixed, as are capital costs, so they are not part of the problem definition.

We applied NSPI to the same weather and pricing data used by [4]. Hourly wind speeds were obtained from the North American Land Data Assimilation Survey for the eight years from January 1, 1998 to December 31, 2005 for three locations: the outer banks of North Carolina (33.9375N, 77.9375W), the lake shore near Cleveland, Ohio (41.8125N, 81.5625W), and the ocean shore near Point Judith, Rhode Island (41.3125N, 71.4375W). Day-ahead hourly prices were obtained from the PJM market for the New Jersey area for the period from January 1, 2002 to December 31, 2009.



(a)

(b)

(c)

(d)

Fig. 4. (a) Price per MWh of electricity from 2002 to 2009, (b) wind speed for North Carolina, (c) wind speed for Rhode Island, (d) wind speed for Ohio

The sequences of prices and wind speeds for the three locations over time are shown in Fig. 4. Each point indicates the wind speed or the price at hour 12 of each day.

For any given day, the action vector is a 24-dimensional vector $(a_1, ..., a_{24})$, where a_h is the amount of electricity promised to be supplied at hour h of the following day. The state at the end of a given day is the vector $(w_1, ..., w_{24}, s_{24}, p_1, ..., p_{24})$ where w_h is the wind speed at hour h of the day, s_{24} is the electricity in storage at the end of the day, and p_h is the per-unit bid price of electricity at hour h of the next day.

Since the storage level varies within a day, we define s_h to be the storage level at the end of hour h for $h = 1$ to $h = 24$. We further define $L_h = L(w_h)$ to be the power generated from the wind speed w_h. To compute L_h we use a numerical approximation of the power generation curve for an industry standard General Electric 1.5MW SL wind turbine. The approximation was graciously provided by Lauren Hannah via email.

For each hour, the excess available electricity compared to the promised supply is $e_h = s_{h-1} + L(w_h) - a_h$. The storage level at the end of each hour is $s_h = \max\{0, \min\{e_h, M\}\}$ where M is the maximum storage capacity. The units purchased on the spot market during each hour are $b_h = \max\{0, -e_h\}$ and the units dumped during each hour are $d_h = \max\{0, e_h - M\}$. The daily revenue is thus

$$R = \sum_{h=1}^{24} p_h a_h - 2p_h b_h - 5d_h.$$

6.1 Applying NSPI

The first stage of applying NSPI is to learn a transition model that can estimate the expected next state $\hat{s}$ based on any current state s and any selected action a. Here, the learned transition model consists of two matrices T_w and T_p that predict the wind velocities and prices seen in the next state as a function of the current state. Note that these speeds and prices are independent of the current action, and of each other. The matrices T_w and T_p are the least squares solutions of the linear systems

$$(w_1, w_2, ..., w_{24}, y_1, y_2, y_3, y_4)\, T_w = (w'_1, w'_2, ..., w'_{24})$$
$$(p_1, p_2, ..., p_{24}, y_1, y_2, y_3, y_4)\, T_p = (p'_1, p'_2, ..., p'_{24})$$

where the prime notation $'$ refers to the following day in the training data. The four y_i are binary indicator variables that describe which season the current day is in. These indicators are informative because wind velocity and electricity prices have seasonal patterns. The two matrices T_w and T_p can be combined into a single transition matrix

$$T = \begin{bmatrix} T_w & 0 \\ 0 & T_p \end{bmatrix}.$$

The transition model must also predict the storage component of $\hat{s}$, i.e., the estimated storage level $\hat{s}_{24}$ at the end of the next day. This is calculated using the equations above from the action vector and the wind speeds predicted using T_w.

The second stage of applying NSPI is to learn an approximate Q function

$$Q(s,a) = Q(\hat{s}) = \sum_{j=1}^{m} \phi_j(\hat{s}) v_j.$$

We define $m = 26$ basis functions as follows. Let d be the day described by $\hat{s}$. The first basis function is the anticipated revenue $\hat{R}$ achieved during day d, the second basis function is the anticipated storage level $\hat{s}_{24}$ at the end of day d, and the remaining basis functions are the 24 estimated prices for day $d + 1$. The trained weights v_j of these basis functions capture the estimated long-term value of the estimated state $\hat{s}$. Formally, $\phi(\hat{s}) = (\hat{R}, \hat{s}_{24}, \hat{p}_1, ..., \hat{p}_{24})$ where $\hat{R}$ and $\hat{s}_{24}$ are estimates that are computed deterministically from $(\hat{w}_1, ..., \hat{w}_{24})$ and $(\hat{p}_1, ..., \hat{p}_{24})$ using the equations above.

During testing, given a state s the recommended action is computed by maximizing $Q(\hat{s})$ using an interior point algorithm [1]. Specifically,

$$a^* = \operatorname*{argmax}_a Q(s,a) = \operatorname*{argmax}_a \sum_{j=1}^{m} \phi_j([s\ a]T) v_j.$$

Table 1. Average annual reward in $1,000 for fixed storage sizes for ADPS and NSPI

Site	Storage capacity	ADPS	NSPI
NC	7.5MWh	114.86	149.80
	15MWh	163.51	184.70
	30MWh	205.38	208.95
OH	7.5MWh	90.53	123.32
	15MWh	131.83	155.67
	30MWh	171.82	181.09
RI	7.5MWh	107.60	138.56
	15MWh	155.00	173.23
	30MWh	200.83	197.75

This maximization is computationally efficient because the 24-dimensional action vector is continuous. With discrete actions, searching the action space would be much more expensive.

6.2 Experiments

Training and test data are identical to those used previously [4]. Each location is an independent agent. The first three years of data (01/01/1998 to 12/31/2000 for wind, 01/01/2002 to 12/31/2004 for prices) are used to create the training samples. The policy for generating training samples is persistent [4]: at the end of each day, commit for the next day as much electricity as was generated (sold, stored, or dumped) on the current day. Based on this commitment, the actual next state and reward are calculated. using the transition equations above. The training policy is followed for each day of the first 3 years, resulting in 1095 samples from the 1096 days. All learning uses a discount factor $\gamma = 0.9$. This numerical value is chosen somewhat arbitrarily, based on the assumption that weather and prices are predictable at most a few days in advance, so decisions need not take into account the far distant future. The strength of regularization was chosen empirically to be $\lambda = 20$.

For testing, each wind farm begins with no stored electricity and is followed over the last 5 years of data. For each wind farm, different policies are separately learned and tested for storage capacities of 7.5MWh, 15MWh and 30MWh. Thus nine policies are learned and evaluated: three for each wind farm, for three different storage capacities.

Table 1 compares the policies learned using NSPI with the policies learned using the Approximate Dynamic Programming for Storage (ADPS) algorithm [4]. The metric used for comparisons is the annual revenue averaged over the five year test period. The paper [4] presents several results obtained using different parameters for the ADPS method. For comparison purposes we use the best performances obtained via ADPS for each case, i.e., each row in the table may use different parameters for ADPS to achieve the best performance. NSPI outperforms ADPS in all but the last case. The biggest improvement is for the smaller storage sizes, for which the task is intrinsically more difficult because a large storage capacity can rarely be used fully. The performance improvement is encouraging considering that ADPS is a method designed specifically for solving storage problems while NSPI is a general method for learning Q functions.

To estimate the transition model, [4] employ a Dirichlet process mixture model, which is considerably more complex than the linear model used here. Another advantage of the approach here is that it treats storage level as a continuous variable, while the ADPS method discretizes the possible storage levels.

The weather and pricing data used here, and in previous research, are from different time periods. This fact implies that wind speeds and prices are statistically independent. In general, these can be correlated. In this case, additional information in the form of tomorrow's prices would be available for predicting tomorrow's wind speeds. We can take advantage of this additional information simply by not restricting the transition matrix T to be block-diagonal. Similarly, if tomorrow's prices are correlated with today's actions by the agent (because other market participants observe these actions and react to them), this can be taken into account by including the action vector when learning T. This will be necessary if the agent produces a significant share of the electricity supply of the region, so that it has market power and cannot be assumed to be merely a price-taker.

7 Discussion

This paper proposes a simple but useful approach to the definition of basis functions for use in a reinforcement learning method. The approach is driven by the observation that the goal of an optimal policy is to choose the best next state. The primary idea is to write $Q(s, a) = Q(\hat{s})$ where $\hat{s}$ is an estimate of the expected next state reached by taking action a in the current state s. The second idea is that basis functions should capture how good the estimated expected next state $\hat{s}$ is. For goal-oriented tasks, the goodness of $\hat{s}$ can be captured by basis functions that represent local neighborhoods, because the coefficient of each basis function can then indicate the proximity of the neighborhood to good and bad terminal states. For tasks such as inventory management, each basis function can describe a certain component or aspect of the following state, such as the level or availability of a particular resource.

The intuitions just described are made concrete in the NSPI algorithm, which is the well-known least squares policy iteration method with the Q function changed to depend on the expected next state $\hat{s}$ instead of on s and a directly. Experimental results demonstrate the effectiveness of NSPI in three domains, including one with high-dimensional continuous state and action spaces.

One issue worth exploring with NSPI is using a more complex but possibly more accurate transition model. The experiments above use a linear transition model, but Gaussian processes, Dirichlet process mixture models, and others have been used effectively in prior work. It is possible that NSPI may perform better with a more accurate transition models, especially in domains where next-state dynamics are highly nonlinear, such as robot navigation with obstacles, including the well-known so-called puddle world. Another avenue for future work is to use methods other than LSPI after the transition model has been learned. In particular, the process suggested here for making the Q function depend on estimated next states could be combined with the fitted Q iteration algorithm, and related methods [3,2].

References

1. Byrd, R.H., Gilbert, J.C., Nocedal, J.: A trust region method based on interior point techniques for nonlinear programming. Mathematical Programming 89, 149–185 (1996)
2. Elkan, C.: Reinforcement Learning with a Bilinear Q Function. In: Sanner, S., Hutter, M. (eds.) EWRL 2011. LNCS, vol. 7188, pp. 78–88. Springer, Heidelberg (2012)
3. Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. Journal of Machine Learning Research 6(1), 503–556 (2005)
4. Hannah, L., Dunson, D.B.: Approximate dynamic programming for storage problems. In: Proceedings of the 28th International Conference on Machine Learning (ICML), pp. 337–344 (2011)
5. Hesami, A.: Matlab implementation of inverted pendulum, http://webdocs.cs.ualberta.ca/~sutton/pole.zip
6. Howard, R.A.: Comments on the origin and application of Markov decision processes. Management Science 14(7), 503–507 (1968)
7. Jong, N., Stone, P.: Model-based function approximation in reinforcement learning. In: Proceedings of the Sixth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 658–665. ACM (2007)
8. Lagoudakis, M.G., Parr, R., Bartlett, L.: Least-squares policy iteration. Journal of Machine Learning Research 4, 1107–1149 (2003)
9. Mahadevan, S., Maggioni, M.: Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. Journal of Machine Learning Research, 2169–2231 (2007)
10. Melo, F.S., Lopes, M.: Fitted Natural Actor-Critic: A New Algorithm for Continuous State-Action MDPs. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 66–81. Springer, Heidelberg (2008)
11. Menache, I., Mannor, S., Shimkin, N.: Basis function adaptation in temporal difference reinforcement learning. Annals of Operations Research 134(1), 215–238 (2005)
12. Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., Littman, M.: An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In: Proceedings of the 25th International Conference on Machine Learning (ICML), pp. 752–759 (2008)
13. Powell, W.B.: Merging AI and OR to solve high-dimensional stochastic optimization problems using approximate dynamic programming. INFORMS Journal on Computing 22(1), 2–17 (2010)
14. Smart, W.D., Kaelbling, L.P.: Practical reinforcement learning in continuous spaces. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 903–910. Morgan Kaufmann (2000)
15. Sutton, R.S.: Reinforcement learning architectures for animats. In: Proceedings of the International Workshop on the Simulation of Adaptive Behavior: From Animals to Animats, pp. 288–296. MIT Press (1991)
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. Cambridge University Press (1998)
17. Tsitsiklis, J.N.: Commentary—perspectives on stochastic optimization over time. INFORMS Journal on Computing 22(1), 18–19 (2010)
18. Uc Cetina, V.: Multilayer perceptrons with radial basis functions as value functions in reinforcement learning. In: Proceedings of the 16th European Symposium on Artificial Neural Networks (ESANN), pp. 161–166 (2008)
19. Wang, H.O., Tanaka, K., Griffin, M.F.: An approach to fuzzy control of nonlinear systems: stability and design issues. IEEE Transactions on Fuzzy Systems 4(1), 14–23 (1996)

Structured Apprenticeship Learning

Abdeslam Boularias[1], Oliver Krömer[2], and Jan Peters[1,2]

[1] Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany
[2] Darmstadt University of Technology, 64289 Darmstadt, Germany

Abstract. We propose a graph-based algorithm for apprenticeship learning when the reward features are noisy. Previous apprenticeship learning techniques learn a reward function by using only local state features. This can be a limitation in practice, as often some features are misspecified or subject to measurement noise. Our graphical framework, inspired from the work on Markov Random Fields, allows to alleviate this problem by propagating information between states, and rewarding policies that choose similar actions in adjacent states. We demonstrate the advantage of the proposed approach on grid-world navigation problems, and on the problem of teaching a robot to grasp novel objects in simulation.

1 Introduction

Programming robots to perform complicated tasks, such as grasping and manipulating objects, is a laborious and time-intensive engineering process. Markov Decision Processes (MDPs) provide an efficient mathematical tool to handle such tasks with minimum human effort. In this framework, the task is simply defined by a reward function. However, in many problems, even the specification of a reward function is not always straightforward. An alternative approach consists of demonstrating examples of a desired behavior and learning a policy that leads to a similar behavior. This type of learning is known as imitation learning and has been widely explored in robotics [1].

Abbeel and Ng [2] introduced a new paradigm of imitation learning known as apprenticeship learning. Rather than directly mimicking the actions of the human, the aim of apprenticeship learning is to recover a reward function under which the human policy is optimal. The learned reward function is then used to find an optimal policy. The process of recovering a reward function is known as Inverse Reinforcement Learning (IRL).

Prior work on apprenticeship learning is based on representing the rewards as a function of state-action features [3–8]. However, this can be a problem in practice when the reward features are noisy or misspecified. Therefore, the features specified by a user are not always sufficient for describing a reward function and for choosing actions accordingly.

An example problem would be planning to grasp an unknown object using visual information. The calculated features are often subject to noise due to measurement errors and self-occlusions. It is also difficult to encode the preference for grasping an object from a specific part, such as a handle, given that these parts come in different shapes.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 227–242, 2012.

A similar problem in computer vision, known as segmentation, has been efficiently solved using a family of graphical models known as Markov Random Fields (MRFs) [9–12]. The key insight behind the performance of Markov fields is that neighbor points on an image tend to have similar labels. Therefore, even a small set of noisy features can be sufficient for classifying a point when considered together with its neighbors. However, Markov fields classify the points by using only immediate costs (or rewards) and cannot be used for learning complex goal-directed behaviors, such as manipulating objects.

In this paper, we build on this insight and introduce a new apprenticeship learning technique that extends Markov Random Fields to sequential decision-making problems. We start by specifying a graph that loosely indicates which pairs of states are supposed to have similar optimal actions. Subsequently, we derive a distribution on policies, wherein the probability of a policy is proportional to its value, and inversely proportional to the number of pairs of adjacent states that have different actions. Consequently, policies are penalized for selecting actions that are inconsistent with the graph. We show that this distribution is an MRF, and describe a dynamic programming procedure that reduces planning in MDPs with MRFs to a sequence of inference problems in MRFs.

The experimental analysis, presented at the end of this paper, shows that this approach can improve the performance of an apprenticeship learning algorithm when the reward features are noisy or misspecified. Specifically, we compare the proposed algorithm to the MaxEnt IRL algorithm [7] on grid-worlds with long planning horizons. We also compare to our previous work on learning to grasp new objects [13]. In [13], the grasping points on an object are classified using an MRF, while the preshaping and the approach direction of the robot hand are given by a heuristic. In this paper, we show how to learn complete grasping policies by using structured apprenticeship learning.

2 Background

In this section, we provide the theoretical background that is necessary for understanding the remainder of this paper.

2.1 Markov Decision Processes

Formally, a Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, T, R, \mu_0, \gamma)$, where $\mathcal{S}$ is a set of states and $\mathcal{A}$ is a set of actions. T is a transition function with $T(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ for $s, s' \in \mathcal{S}, a \in A$, and R is a reward function where $R(s, a)$ is the reward given for executing action a in state s. The initial state distribution is denoted by μ_0, and $\gamma \in [0, 1]$ is a discount factor. A Markov Decision Process without a reward function is denoted by MDP\R. We assume that the reward function is a linear combination of K feature vectors ϕ_k with weights θ_k,

$$\forall (s, a) \in \mathcal{S} \times \mathcal{A} : R(s, a) = \sum_{k=1}^{K} \theta_k \phi_k(s, a).$$

A deterministic policy π is a function that returns an action $a = \pi(s)$ for each state s. The expected return $J(\pi)$ of a policy π is the expected sum of rewards that will be received when following policy π, i.e.

$$J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | \mu_0, \pi, T].$$

An optimal policy π^* is one satisfying $\pi^* \in \arg\max_\pi J(\pi)$. The expectation of a feature ϕ_k for a policy π is defined as

$$\phi_k^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi_k(s_t, a_t) | \mu_0, \pi, T].$$

Using this definition, the expected return of a policy π can be written as a linear function of the feature expectations

$$J(\pi) = \sum_{k=1}^{K} \theta_k \phi_k^\pi.$$

2.2 Apprenticeship Learning

The aim of apprenticeship learning is to find a policy π that is nearly as good as a policy π^E demonstrated by a human expert, i.e., $J(\pi) \geq J(\pi^E) - \epsilon$. However, the expected returns of π and π^E cannot be directly compared, unless a reward function is provided. As a solution to this problem, Ng and Russell [14] proposed to first learn a reward function, assuming that the expert is optimal, and then use it to recover the expert's generalized policy.

However, the problem of learning a reward function given an optimal policy is ill-posed [2]. In fact, a large class of reward functions may lead to the same optimal policy. Most of the apprenticeship learning literature has focused on solving this particular problem. Examples of the proposed solutions include incorporating prior information on the reward function, minimizing the margin $\|J(\pi) - J(\pi^E)\|$, or maximizing the entropy of the distribution on state-actions under a learned stochastic policy [7]. In this work, we will use the maximum entropy regularization.

The principle of maximum entropy states that the simplest policy that best represents the provided examples is the one with the highest entropy, subject to the constraint of matching the expected return of the demonstrated actions. This latter constraint can be satisfied by ensuring that the feature counts of the learned policy match with those of the demonstration,

$$\forall k \in \{1, \ldots, K\} : \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi_k(s_t, a_t) | \mu_0, \pi, T] = \hat{\phi}_k \tag{1}$$

where $\hat{\phi}_k$ denotes the empirical expectation of feature k calculated from the demonstration. The MaxEnt IRL approach [7] consists of finding the parameters θ of a policy π that maximizes the entropy of the distribution on the state-action trajectories subject to constraint (1). Solving this problem leads to maximizing the likelihood of the demonstrated trajectories under an exponential distribution of the policies.

2.3 Markov Random Fields

A Markov Random Field (MRF) is a graphical model used for representing joint probability distributions. The MRF defines a probability distribution over N discrete variables $Y = \{y_1, \dots, y_n\}$. Each variable corresponds to the label of a node in a graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges. Each node x_i is assigned to a label y_i from a set $\mathcal{L}$ of possible labels. Therefore, the MRF defines a probability distribution over $\mathcal{L}^N$.

We focus on a particular tractable class of MRFs known as Associative Markov Network (AMN) [15], where potentials $\rho(x_i, y_i)$ and $\rho(x_i, x_j)$ are associated with each node $x_i \in \mathcal{V}$ labeled by y_i, and each edge $(x_i, x_j) \in \mathcal{E}$ such that x_i and x_j have the same label. The AMN model uses the log-linear function for representing a potential as a function of the features, i.e. $\log \rho(x_i, y_i) = \sum_k \theta_k \phi(x_i, y_i)$ and $\log \rho(x_i, x_j) = \sum_k \lambda_k \psi_k(x_i, x_j)$, where $\theta_k \in \mathbb{R}$ are node weights, $\phi_k(x_i, y_i) \in \mathbb{R}$ are features of node x_i labeled by y_i, $\lambda_k \in \mathbb{R}$ are edge weights, and $\psi_k(x_i, x_j) \in \mathbb{R}$ are features of edge (x_i, x_j). The joint probability distribution on the labels $(y_1, \dots, y_n)$ is given by

$$P(y_1, \dots, y_n | x_1, \dots, x_n) \propto \exp\Big(\sum_{x_i \in \mathcal{V}} \sum_k \theta_k \phi(x_i, y_i) + \sum_{\substack{(x_i, x_j) \in \mathcal{E} \\ \text{s.t. } y_i = y_j}} \sum_k \lambda_k \psi_k(x_i, x_j) \Big).$$

3 Structured Apprenticeship Learning

In this section, we present the structured apprenticeship learning problem and show how it can be solved efficiently.

3.1 Key Insight

The classical framework of apprenticeship learning is based on hand-coding the features ϕ of the reward and learning the weights θ. In practice, it is often difficult to find an appropriate set of features that contains necessary and sufficient information about the reward. Moreover, these features are usually obtained from empirical data and are subject to measurement errors. On the other hand, most real-world problems exhibit a certain structure wherein states that are close together tend to have the same optimal action. This implicit information about the optimal policy, given by the structure, can be used to partially overcome the problem of finding an appropriate set of reward features. Intuitively, a distance can be interpreted as a kernel that measures the similarity between the approximate values of two states under an optimal policy. In many robotic applications, such as navigation, this distance is simply the Euclidean or the geodesic distance.

3.2 Problem Statement

Given an appropriate definition of a measure between states, we construct a k-nearest neighbors graph where the nodes correspond to states and the set of edges is denoted by $\mathcal{E}$. Structured apprenticeship learning can be formulated as the problem of finding a distribution P on deterministic policies, denoted by π, that has the highest possible entropy while matching the expected value of each state-action feature ϕ_k with the one calculated from the demonstration, $\hat{\phi}_k$. Moreover, a set of edge features ψ_k is defined over edges in $\mathcal{E}$. Edge features $\psi_k(s_i, s_j)$ take a zero value when $\pi(s_i) \neq \pi(s_j)$. The distribution P should also match the expected value of each edge feature ψ_k with the one calculated from the demonstration, $\hat{\psi}_k$. Thus, this problem can be defined as

$$\max_{P,\mu^{\pi}} \Big(- \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) \log P(\pi) \Big), \tag{2}$$

$$\text{subject to}$$

$$\sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) = 1,$$

$$\forall \phi_k : \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) \sum_{s \in \mathcal{S}} \mu^{\pi}(s) \phi_k(s, \pi(s)) = \hat{\phi}_k,$$

$$\forall \psi_k : \sum_{(s_i, s_j) \in \mathcal{E}} \psi_k(s_i, s_j) \sum_{\pi, \pi(s_i) = \pi(s_j)} P(\pi) = \hat{\psi}_k,$$

where $\mu^{\pi}(s)$ is the expected discounted number of visits of state s,

$$\forall \pi, s : \mu^{\pi}(s) = \mu_0(s) + \gamma \sum_{s'} \mu^{\pi}(s') T(s', \pi(s'), s).$$

The state-action features ϕ_k correspond to the basis functions of the reward function. The edge features ψ_k can be interpreted as the basis functions of a reward given for choosing the same actions for adjacent states. For instance, $\psi_k(s_i, s_j)$ can be the inverted distance between states s_i and s_j.

Another interesting example is when $\psi_k(s_i, s_j) = 1, \forall (s_i, s_j) \in \mathcal{E}$, in which case the last constraint in Problem 2 corresponds to forcing the probability of the policies that choose the same action in adjacent states to be equal to the empirical probability $\hat{\psi}_k$. In this manner, the robot learns a policy that also imitates the structure of the human policy. If the provided structure is not relevant to the task, then $\hat{\psi}_k$ will be low. Consequently, the learned policy will not be forced to have a similar structure.

3.3 Proposed Solution

The Lagrangian of this problem is given by

$$\begin{aligned} L(P,\eta,\theta,\lambda) = &- \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi)\log P(\pi) + \eta\Big(\sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) - 1\Big) \\ &+ \sum_k \theta_k \Big(\sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) \sum_{s \in \mathcal{S}} \mu^\pi(s)\phi_k(s,\pi(s)) - \hat{\phi}_k \Big) \\ &+ \sum_k \lambda_k \Big(\sum_{\pi,\pi(s_i)=\pi(s_j)} P(\pi) \sum_{(s_i,s_j)\in\mathcal{E}} \psi_k(s_i,s_j) - \hat{\psi}_k \Big). \end{aligned}$$

Therefore

$$\begin{aligned} \partial_{P(\pi)} L(P,\eta,\theta,\lambda) = &\sum_s \mu^\pi(s) \sum_k \theta_k \phi_k(s,\pi(s)) + \sum_{\substack{(s_i,s_j)\in\mathcal{E} \\ \text{s.t. } \pi(s_i)=\pi(s_j)}} \sum_k \lambda_k \psi_k(s_i,s_j) \\ &- \log P(\pi) + \eta - 1. \end{aligned}$$

The optimal solution satisfies $\partial_{P(\pi)} L(P,\eta,\theta,\lambda) = 0$, then

$$\log P(\pi) = \sum_s \mu^\pi(s) \sum_k \theta_k \phi_k(s,\pi(s)) + \sum_{\substack{(s_i,s_j)\in\mathcal{E} \\ \text{s.t. } \pi(s_i)=\pi(s_j)}} \sum_k \lambda_k \psi_k(s_i,s_j) - \log Z(\theta,\lambda), \quad (3)$$

where $Z(\theta,\lambda)$ is a partition function. Therefore

$$P(\pi) \propto \exp\Big(\sum_s \mu^\pi(s) \sum_k \theta_k \phi_k(s,\pi(s)) + \sum_{\substack{(s_i,s_j)\in\mathcal{E} \\ \text{s.t. } \pi(s_i)=\pi(s_j)}} \sum_k \lambda_k \psi_k(s_i,s_j) \Big). \quad (4)$$

3.4 Relation to Other Methods

The probability distribution given by Equation 4 is a Markov Random Field, as illustrated in Figure 1. The probability of choosing action a in a given state s depends on the expected return of (s,a) and the actions chosen in neighboring states. There is a clear similarity between the distribution of policies in structured apprenticeship learning and the distribution of joint labels in Associative Markov Networks (AMN) [16] in particular. In fact, the proposed framework generalizes AMN to sequential decision making problems. States are the input points and actions are the labels, the labeling cost is given by the reward. The label distribution in AMN can be derived from Equation 4 by setting $\gamma = 0$,

Table 1. Relation between Structured Apprenticeship Learning and other methods

	$\lvert\mathcal{E}\rvert = 0$	$\lvert\mathcal{E}\rvert \neq 0$
$\gamma = 0$	Logistic regression	AMN [16]
$\gamma \neq 0$	MaxEnt IRL [7]	Structured Apprenticeship Learning

then μ^π becomes equal to μ_0, the initial state distribution, which is a constant factor that can be included in the features ϕ_k. Also, the MaxEnt method [7] can be derived from Equation 4 by setting $\lambda = 0$. Note that Ziebart et al. [7] use a distribution on paths instead of policies. The following table shows the relation between structured apprenticeship learning and other methods.

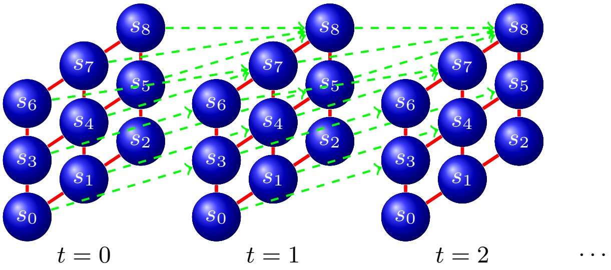


Fig. 1. Structured Markov Decision Process

Finally, we should mention that the concept of using structured output prediction for learning by demonstration was also considered in [17]. Although, the approach of [17] consists in classifying robot trajectories using conditional random fields in a supervised learning fashion, without considering a reward function and planning the trajectories.

3.5 Learning Procedure

We provide a solution for finding the reward parameters θ and the structure parameters λ that maximize the likelihood of an expert policy π^E demonstrated in a domain that can be different from the testing domain. The demonstrations are given by state-action trajectories, and the empirical feature averages $\hat{\phi}_k$ and $\hat{\psi}_k$ are calculated from these trajectories. We denote the set of states that appear in the demonstrations by $\mathcal{S}^E$, and the corresponding set of edges by $\mathcal{E}^E$.

From the Representer Theorem [18], we know that the parameters θ and λ that maximize $\log P(\pi)$ (Equation 3) are given by

$$\theta_k = \sum_{s \in \mathcal{S}^E} \alpha_s \phi_k(s, \pi^E(s)),$$

$$\lambda_k = \sum_{\substack{(s_i,s_j)\in\mathcal{E}^E \\ \text{s.t. } \pi^E(s_i)=\pi^E(s_j)}} \beta_{s_i,s_j} \psi_k(s_i, s_j),$$

where $\alpha_s, \beta_{s_i,s_j} \in \mathbb{R}$. Therefore, the log-probability of a deterministic policy π defined on an arbitrary structured MDP $(\mathcal{S}, \mathcal{E}, \mathcal{A}, T, \mu_0, \gamma)$ is given by

$$\log P(\pi) = \underbrace{R_\beta(\mathcal{E}, \pi)}_{\text{structure reward}} + \underbrace{\sum_s \mu^\pi(s) R_\alpha(s, \pi(s))}_{\text{expected return}} - \log Z(\alpha, \beta), \tag{5}$$

$$R_\alpha(s, a) \overset{def}{=} \sum_{s' \in \mathcal{S}^E} \alpha_{s'} k(\langle s, a\rangle, \langle s', \pi^E(s')\rangle),$$

$$R_\beta(\mathcal{E}, \pi) \overset{def}{=} \sum_{\substack{(s_i,s_j)\in\mathcal{E} \\ \pi(s_i)=\pi(s_j)}} \sum_{\substack{(s'_i,s'_j)\in\mathcal{E}^E \\ \pi^E(s'_i)=\pi^E(s'_j)}} \beta_{s'_i,s'_j} k_e(\langle s_i, s_j\rangle, \langle s'_i, s'_j\rangle),$$

where k and k_e are kernel functions used for measuring similarities between state-action couples and between edges respectively, they are defined as

$$k(\langle s, a\rangle, \langle s', a'\rangle) = \sum_k \phi_k(s, a)\phi_k(s', a'),$$

$$k_e(\langle s_i, s_j\rangle, \langle s'_i, s'_j\rangle) = \sum_k \psi_k(s_i, s_j)\psi_k(s'_i, s'_j).$$

The partition function $Z(\alpha, \beta)$ is given by

$$Z(\alpha, \beta) = \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} \exp\Big(R_\beta(\mathcal{E}, \pi) + \sum_s \mu^\pi(s) R_\alpha(s, \pi(s))\Big). \tag{6}$$

In the learning phase, Equation 5 is used for finding parameters α and β that maximize the likelihood of the expert's policy π^E. Since this likelihood function is concave, the optimal parameters α and β can be found by using standard optimization methods, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. A key drawback of this approach is the computational cost of calculating the partition function $Z(\alpha, \beta)$ at each step of the optimization algorithm, which is $\mathcal{O}(|\mathcal{A}|^{|\mathcal{S}|}|\mathcal{S}|^3)$, this corresponds to the cost of calculating the values of all the deterministic policies using value iteration for instance.

In practice, this problem can be addressed by using several possible tricks. First, we reuse the values calculated for a given policy π as the initial values of all the policies that differ from π in one state only, the values of these policies are found after a few iterations. Second, we may be interested in finding a probability distribution on a restricted set of eligible policies, instead of all possible policies. The learned reward function will then discriminate the expert's policy π^E against other alternative candidates. For example, we can consider all the policies that differ from π^E by only one action, or the optimal policies given a sequence of hypothesized reward functions, as in Maximum Margin Markov Networks [16]. Finally, we can decompose the state space into a set of weakly connected components $\mathcal{C} = \{S_i \subset \mathcal{S}\}$, and separately calculate the partition function of each component, which reduces the computational complexity to $\mathcal{O}(\sum_{\mathcal{S}_i \in \mathcal{C}} |\mathcal{A}|^{|\mathcal{S}_i|}|\mathcal{S}_i|^3)$. This procedure is given in Algorithm 1.

```
Input: A structured MDP\R $(\mathcal{S}, \mathcal{E}, \mathcal{A}, T, \mu_0, \gamma)$ ;
Let $\mathcal{C}_0$ be the set of weakly connected components in the graph defined by the
states and the edges in $\mathcal{E}$ ;
$t \leftarrow 0$;
repeat
    $t \leftarrow t + 1$ ; $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}$;
    foreach $\mathcal{S}_i \in \mathcal{C}_t, s \in \mathcal{S}_i, a \in \mathcal{A}, \mathcal{S}_j \in \mathcal{C}_t, s' \in \mathcal{S}_j$ do
        if $T(s, a, s') \neq 0$ then
            $\mathcal{S}_k = \mathcal{S}_i \cup \mathcal{S}_j$ ; $\mathcal{C}_t \leftarrow \mathcal{C}_t \cup \{\mathcal{S}_k\}$ ;
        end
    end
until $\mathcal{C}_t = \mathcal{C}_{t-1}$;
Output: A set of weakly connected components $\mathcal{C}$;
```

Algorithm 1. Decomposing the state space into weakly connected components

3.6 Finding Policies for Finite Horizons

For a finite horizon H, an optimal policy π can be non-stationary, i.e. $\pi_{0:H} = (\pi_0, \pi_1, \dots, \pi_H)$. Moreover, if the initial state distribution μ_0 is unknown, then Problem 2 should be stated for every initial state and time-step. Assuming that the reward and structure (edges) parameters α and β are stationary (they do not depend on the starting state or time), the solution is given by the conditional probability distribution

$$P(\pi_t|\pi_{t+1:H}) \propto \exp\Big(R_\beta(\mathcal{E}, \pi_t) + \sum_s V_\alpha^{\pi_{t:H}}(s)\Big),$$

$$V_\alpha^{\pi_{t:H}}(s) = R_\alpha(s, \pi_t(s)) + \gamma \sum_{s'} T(s, \pi_t(s), s') V_\alpha^{\pi_{t+1:H}}(s').$$

Algorithm 2 describes a dynamic programming procedure for finding a policy $\pi^*_{0:H} = (\pi^*_0, \pi^*_1, \dots, \pi^*_H)$ that satisfies

$$\forall t \in [0, H] : \pi^*_t = \operatorname*{argmax}_{\pi_t \in \mathcal{A}^{|\mathcal{S}|}} P(\pi_t|\pi^*_{t+1:H}).$$

The planning problem is reduced to a sequence of inference problems in Markov fields, i.e. finding joint labels that have the highest probability. The inference problem itself also can be efficiently solved using techniques such as graph min-cut [19], α-expansions and linear programming relaxation [15]. We adopt the α-expansions technique. For more details, we refer the reader to Taskar (2004) [16].

4 Experiments

In this section, we present experiments on learning to navigate in gridworlds with noisy reward features, and learning to grasp unknown objects. We compare

```
Input: A structured MDP $(\mathcal{S}, \mathcal{E}, \mathcal{A}, T, \gamma)$ with reward parameters $\alpha$ and
       structure parameters $\beta$, a planning horizon $H$;
$\forall s \in \mathcal{S}, \forall a \in \mathcal{A} : Q_\alpha^{H+1}(s,a) = 0$;
for $t = H : 0$ do
    $\forall s \in \mathcal{S}, \forall a \in \mathcal{A} : Q_\alpha^t(s,a) = R_\alpha(s,a) + \gamma \sum_{s'} T(s,a,s') Q_\alpha^{t+1}(s', \pi_{t+1}^*(s'))$;
    Use an inference algorithm in the MRF defined on the graph $(\mathcal{S}, \mathcal{E})$ to label
    states with actions. The potential of the nodes is given by $Q_\alpha$ and the
    potential of the edges is given by $R_\beta$;
    Denote by $\pi_t^*$ the policy returned by the inference algorithm;
end
Output: A non-stationary policy $\pi^* = (\pi_0^*, \pi_1^*, \ldots, \pi_H^*)$;
```

Algorithm 2. Reducing planning in structured MDP to inference in MRFs

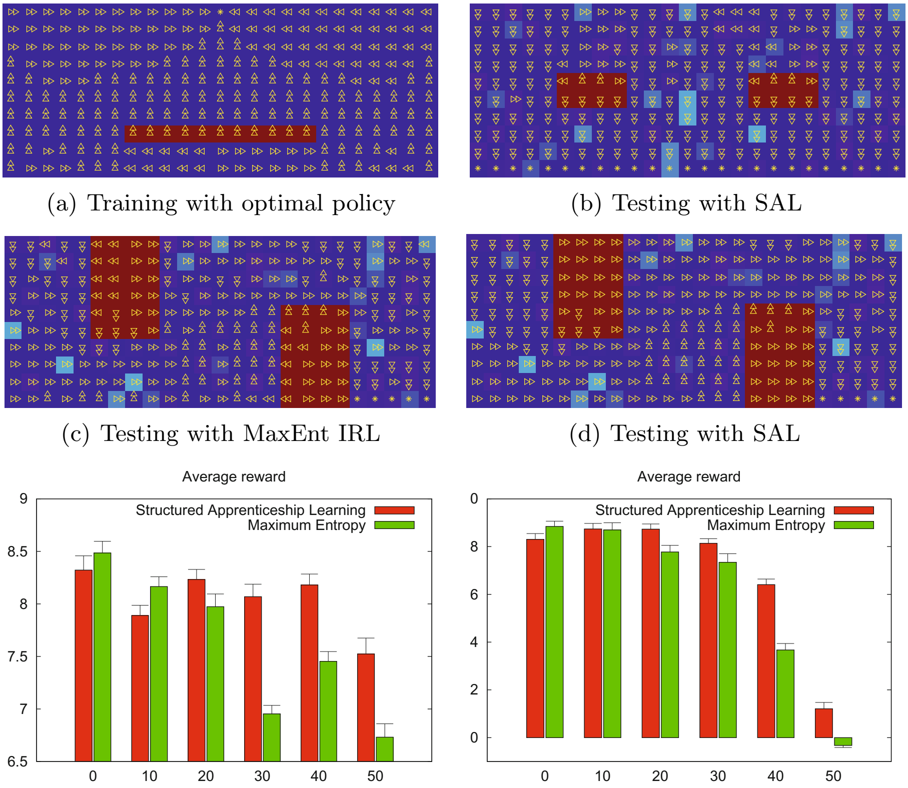
Structured Apprenticeship Learning (SAL) with MaxEnt IRL [7]. Note that Ziebart et al. [7] used a distribution on trajectories while we use distributions on policies.

4.1 Gridworlds

We consider 10 × 25 gridworlds. A state corresponds to the location of a robot, which has four actions for moving in one of the four directions of the compass. The actions are stochastic, the robot is randomly moved to one of the four adjacent states with probability 5%. There are two reward features, ϕ_1 and ϕ_2. Feature ϕ_1 takes value 1 in the goal states and 0 elsewhere. Feature ϕ_2 indicates bad regions that should be avoided (big obstacles, large holes, slippery floors, etc...). The true reward function is given by $R(s,a) = \phi_1(s) - 20\phi_2(s)$.

Based on the assumption that bad states tend to be regrouped in large regions, an optimal policy is expected to select the same action for most of the time, and occasionally turn left or right to avoid an obstacle. Therefore, we use the Manhattan distance on the grid as a similarity measure between states, and consider the immediate neighbors as adjacent states in the graph used by SAL. We use a constant edge feature, set to 1. The parameters of MaxEnt IRL and SAL are learned by maximizing the likelihood of the policy shown in Figure 2 (a) using the BFGS method. In the learning process, we restrict the policy distribution to the set of policies that differ from the demonstration in at most one state. This was sufficient for learning fairly accurate reward weights, $(1.2, -18.0)$ for SAL and $(1.2, -20.7)$ for MaxEnt, while reducing the learning time to 173 seconds for SAL and only 4 seconds for MaxEnt. The learned edge weight for SAL is $\lambda = 0.28$.

Tests were performed in two gridworlds, shown in Figures 2 (b,c,d). In each gridworld, there are two large regions characterized by ϕ_2 set to 1. The remaining states have ϕ_2 set to 0. Randomly selected states have been noised, i.e. their values of ϕ_2 were altered to values uniformly sampled from the interval $[0, 0.2]$. We used the α-expansions algorithm for inference in SAL. The average planning times were 0.58 and 1.7 seconds for SAL, and 0.4 and 1.05 seconds for MaxEnt.



(a) Training with optimal policy

(b) Testing with SAL

(c) Testing with MaxEnt IRL

(d) Testing with SAL

(e) First domain results

(f) Second domain results

Fig. 2. Experiments in gridworlds. Blue indicates a low value of a feature associated with negative reward, and red indicates a higher value of that feature. In the testing domains, a white noise is added to the negative-weighted feature of randomly chosen states. Subfigures (e) and (f) show the average rewards of MaxEnt and SAL as a function of the percentage of noisy states.

Figures 2 (e,f) show the average actual rewards of SAL and MaxEnt policies as a function of the percentage of states that have been noised. The results are averaged over 10^5 trials of length 50 with the discount factor $\gamma = 1$. Notice that with low levels of noise, MaxEnt slightly outperforms SAL. This is due to the fact that the SAL policy prefers to choose a similar action for adjacent states, even when sometimes a different action is optimal. For the same reason, SAL significantly outperforms MaxEnt in high levels of noise, where the robot with MaxEnt policy spends most of its time trying to avoid most of the noisy states, as shown in Figure 2 (c).

This experiment shows that SAL can improve over MaxEnt IRL when the reward features are noisy. However, when the noise is too small or absent, the performance of SAL can be lower than that of MaxEnt IRL. This is due to the bias introduced by SAL, which favors smooth policies. Nevertheless, SAL is intended to be used only when the noise level is significant.

4.2 Grasping Unknown Objects

From a high-level point of view, grasping an object can be seen as an MDP with three steps: reaching, preshaping, and grasping (Figure 3). At any step, the robot can either proceed to the next step or restart from the beginning and get a reward of 0. The robot always starts at $t = 0$ from the same initial state s_0, the set of actions corresponds to the set of points on the surface of the object, assuming that the approach direction is always given by the surface normal vector. At $t = 1$, the state is given by a surface point and an approach direction, the set of actions correspond to the set of all possible hand orientations. At $t = 2$, the state is given by a surface point, an approach direction and a hand orientation. There are two possible last actions, closing the fingers or restarting.

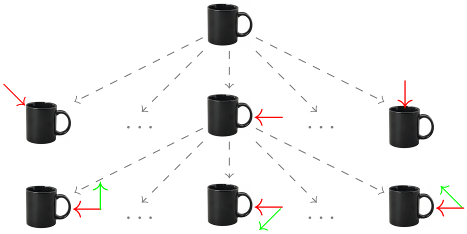


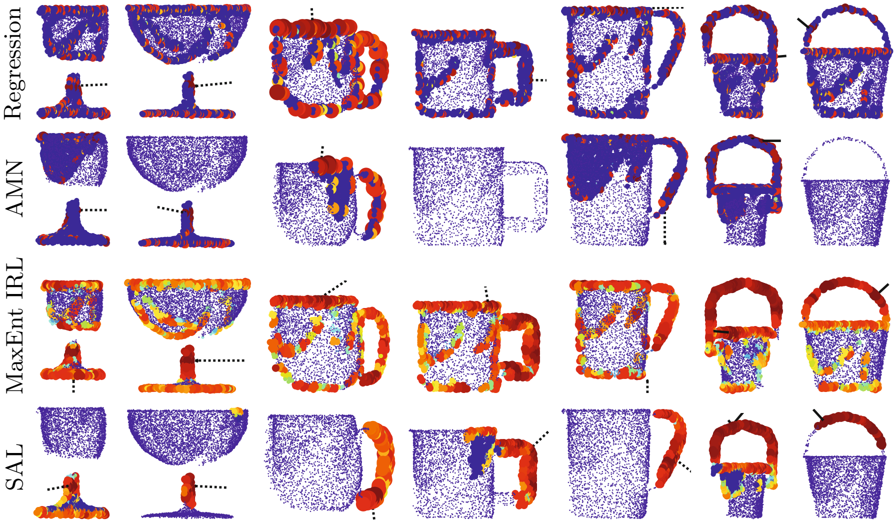
Fig. 3. Grasping as a three-step Markov Decision Process

In this experiment, we are interested in learning to grasp objects from their handles. The reward of each step depends on the current state. There is no reward at $t = 0$. The reward R_1 defined at $t = 1$ is a function of the first three eigenvalues of the scatter matrix defined by the 3D coordinates of the points inside a small ball centered on the selected point. These features indicate if a point is on a handle. The reward R_2, defined at $t = 2$, is a function of collision features. We simulate the trajectories of 10 equidistant points on each finger of a Barrett robot hand (a three-fingered gripper). The collision features are binary variables indicating whether or not there will be a collision with the object, during the grasping, for each one of the specified finger points.

Based on the assumption that points that are close to each other should have the same action (i.e. same approach direction and hand orientation), the structure of this MDP is given by the k-nearest neighbors graph, using the Euclidean distance and $k = 6$ in the state space of positions (or surface points), and the angular distance, with $k = 2$ in the discretized state space of hand orientations. We use a quadratic kernel for learning R_1, and the Hamming distance between the feature vectors as a kernel for learning R_2. We also use a single constant feature for all the edges.

We used one object for training and provided six trajectories leading to a successful grasp from its handle. For testing, we compared SAL with MaxEnt IRL, AMN and Logistic Regression, which is equivalent to AMN without the graph structure. For AMN and Logistic Regression, only the reward R_1 at

Table 2. Learned Q-values at $t = 0$. Each point on an object corresponds to an action. Blue indicates low values and red indicates high values. The black arrow indicates the approach direction in the optimal policy according to the learned reward function.



time-step 1 is learned, since these are classification methods and do not consider subsequent rewards.

Table 2 shows the Q-values at $t = 0$ and the approach directions at optimal grasping points given the reward functions learned by different algorithms. Notice how SAL improves over the other methods by generally giving high values to handle points only. The values of the other points are zeros because the optimal action at these points is to restart rather than to grasp. The confusion in the other methods comes from noise features and self-occlusions. Notice also that SAL improves over AMN by considering the reward at $t = 2$ while making a decision at $t = 1$. Figure 4 shows the percentage of successful grasps using the objects in Table 2. A grasp is labeled successful if it is located on a handle and the hand orientation is orthogonal to the handle and the approach direction.

Note that Ratliff [20] solved a similar grasp prediction problem by using a structured output prediction technique. The experimental setup used in [20] is different from ours since it contained complete 3D models of the objects instead of point clouds. We compared SAL only with MaxEnt IRL, which is a special case of SAL, in order to illustrate the advantage of using MRF. The approach of Ratliff [20] for grasping can be extended in a similar way to handle sequential decision-making. In fact, structured apprenticeship learning with MRFs is one way of using structured prediction in MDPs, one can also use other techniques such as Max-Margin Markov Networks [16].

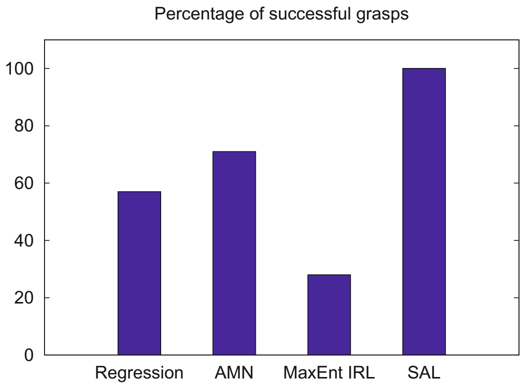


Fig. 4. Percentage of grasps labeled as successful

We also compared with the classification methods used in [13] (AMN and logistic regression) in order to show the advantage of taking future rewards into account in grasping. In fact, AMN and logistic regression use only the reward function at the first time-step for selecting the grasping point, the approach direction is found by using a heuristic. Table 2 clearly shows that structured apprenticeship learning improves over AMN and logistic regression. In fact, the dynamic programming procedure used in SAL (Algorithm 2) backpropagates the cost related to the collisions of the fingers with the object to the first time-step. Therefore, most of the points with a high value are located on handles.

5 Conclusion

Robotic tasks, such as grasping objects, are often difficult to solve by using manual programming. A solution to this problem, known as apprenticeship learning, consists of providing the robot with a demonstration of an optimal policy for solving the task. The robot learns a reward function that explains the demonstration and uses it for generalization.

In this paper, we showed that the reward function alone is not always sufficient for properly explaining a behavior when some features are noisy, or are poorly specified. To solve this problem, we presented a new technique that we called Structured Apprenticeship Learning, and which is inspired by Markov fields. Experiments on navigation and grasping tasks confirmed that structured apprenticeship learning improves over unstructured methods when the features are noisy.

However, our approach suffers from a higher computational complexity compared to standard MDP algorithms. This problem will be investigated in a future work by using well-known approximate inference techniques in graphical models.

References

1. Schaal, S.: Is Imitation Learning the Route to Humanoid Robots? Trends in Cognitive Sciences 3(6), 233–242 (1999)
2. Abbeel, P., Ng, A.Y.: Apprenticeship Learning via Inverse Reinforcement Learning. In: Proceedings of the Twenty-first International Conference on Machine Learning, ICML 2004, pp. 1–8 (2004)
3. Ratliff, N., Bagnell, J., Zinkevich, M.: Maximum Margin Planning. In: Proceedings of the Twenty-Third International Conference on Machine Learning, ICML 2006, pp. 729–736 (2006)
4. Ramachandran, D., Amir, E.: Bayesian Inverse Reinforcement Learning. In: Proceedings of The Twentieth International Joint Conference on Artificial Intelligence, IJCAI 2007, pp. 2586–2591 (2007)
5. Syed, U., Schapire, R.: A Game-Theoretic Approach to Apprenticeship Learning. In: Advances in Neural Information Processing Systems 20, NIPS 2008, pp. 1449–1456 (2008)
6. Syed, U., Bowling, M., Schapire, R.E.: Apprenticeship Learning using Linear Programming. In: Proceedings of the Twenty-Fifth International Conference on Machine Learning, ICML 2008, pp. 1032–1039 (2008)
7. Ziebart, B., Maas, A., Bagnell, A., Dey, A.: Maximum Entropy Inverse Reinforcement Learning. In: Proceedings of The Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, pp. 1433–1438 (2008)
8. Ziebart, B., Bagnell, A., Dey, A.: Modeling Interaction via the Principle of Maximum Causal Entropy. In: Proceedings of the Twenty-Seventh International Conference on Machine Learning, ICML 2010, pp. 1255–1262 (2010)
9. Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., Ng, A.: Discriminative learning of Markov random fields for segmentation of 3d scan data. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR 2005, pp. 169–176 (2005)
10. Munoz, D., Vandapel, N., Hebert, M.: Onboard contextual classification of 3-D point clouds with learned high-order Markov random fields. In: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, ICRA 2009 (2009)
11. Kohli, P., Kumar, P., Torr, P.: P3 and beyond: Solving energies with higher order cliques. In: IEEE International Conference on Computer Vision and Pattern Recognition, ICCVPR 2007 (2007)
12. Ratliff, N., Bagnell, D., Zinkevich, M.: Online subgradient methods for structured prediction. In: Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007 (2007)
13. Boularias, A., Kroemer, O., Peters, J.: Learning Robot Grasping from 3-D Images with Markov Random Fields. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011 (2011)
14. Ng, A., Russell, S.: Algorithms for Inverse Reinforcement Learning. In: Proceedings of the Seventeenth International Conference on Machine Learning, ICML 2000, pp. 663–670 (2000)
15. Taskar, B., Chatalbashev, V., Koller, D.: Learning associative markov networks. In: Proceedings of the Twenty-First International Conference on Machine Learning, ICML 2004 (2004)
16. Taskar, B.: Learning Structured Prediction Models: A Large Margin Approach. PhD thesis, Stanford University, CA (2004)

17. Vakanski, A., Janabi-Sharifi, F., Mantegh, I., Irish, A.: Trajectory learning based on conditional random fields for robot programming by demonstration. In: Proceedings of the IASTED International Conference on Robotics and Applications, RA 2010 (2010)
18. Schölkopf, B., Herbrich, R., Smola, A.J.: A Generalized Representer Theorem. In: Helmbold, D.P., Williamson, B. (eds.) COLT 2001 and EuroCOLT 2001. LNCS (LNAI), vol. 2111, pp. 416–426. Springer, Heidelberg (2001)
19. Boykov, Y., Veksler, O., Zabih, R.: Fast Approximate Energy Minimization via Graph Cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 2001 (1999)
20. Ratliff, N.: Learning to Search: Structured Prediction Techniques for Imitation Learning. PhD thesis, Carnegie Mellon University (2009)

A Bayesian Approach for Classification Rule Mining in Quantitative Databases

Dominique Gay and Marc Boullé

Orange Labs
2, avenue Pierre Marzin, F-22307 Lannion Cedex, France
firstname.name@orange.com

Abstract. We suggest a new framework for classification rule mining in quantitative data sets founded on Bayes theory – without univariate preprocessing of attributes. We introduce a space of rule models and a prior distribution defined on this model space. As a result, we obtain the definition of a parameter-free criterion for classification rules. We show that the new criterion identifies interesting classification rules while being highly resilient to spurious patterns. We develop a new parameter-free algorithm to mine locally optimal classification rules efficiently. The mined rules are directly used as new features in a classification process based on a selective naive Bayes classifier. The resulting classifier demonstrates higher inductive performance than state-of-the-art rule-based classifiers.

1 Introduction

The popularity of association rules [1] is probably due to their simple and interpretable form. That is why they received a lot of attention in the recent decades. E.g., when considering Boolean datasets, an association rule is an expression of the form $\pi : X \to Y$ where the body X and the consequent Y are subsets of Boolean attributes. It can be interpreted as: "*when attributes of X are observed, then attributes of Y are often observed*". The strength of a rule pattern lies in its inductive inference power: from now on, if we observe the attributes of X then we may rely on observing attributes of Y. When Y is a class attribute, we talk about classification rules (like $X \to c$) which seems to be helpful for classification tasks – indeed, "if an object is described by attributes of X then it probably belongs to class c". A lot of efforts have been devoted to this area in the past years and have given rise to several rule-based classifiers (see pioneering work: "Classification Based on Associations" (CBA [22]). Nowadays, there exist numerous CBA-like classifiers which process may be roughly summarized in two steps: *(i)* mining a rule set w.r.t. an interestingness measure, *(ii)* building a classifier with a selected subset of the mined rules (see [7] for a recent well-structured survey). Another research stream exploits a rule induction scheme: each rule is greedily extended using various heuristics (like e.g. information gain) and the rule set is built using a sequential database covering strategy. Following this framework, several rule-induction-based classification algorithms have been

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 243–259, 2012.

proposed; e.g. [9,24,32]. Rule mining and rule-based classification are still ongoing research topics. To motivate our approach, we highlight some weaknesses of existing methods.

Motivation. Real-world data sets are made of quantitative attributes (i.e. numerical and/or categorical). Usually, each numerical attribute is discretized using a supervised univariate method and each resulting interval is then mapped to a Boolean attribute (see section 5 for further related work about mining quantitative data sets). The simple XOR example (see figure 1) shows the limit of such preprocessing. Indeed, it seems there is no valuable univariate discretization for attribute X (resp. Y), thus both attributes might be pruned during preprocessing. If X and Y are individually non-informative, their combination could be class-discriminant: e.g., the rule $(X < x_b) \wedge (Y < y_b) \rightarrow c_2$ is clearly an interesting pattern. Notice that univariate preprocessing of a categorical attribute, like supervised value grouping, is subject to the same drawback.

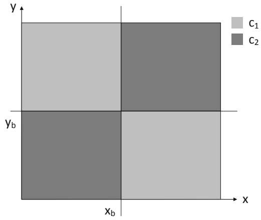


Fig. 1. 2-class XOR data

Another weakness of CBA-like classifiers is parameter tuning. Most of the methods works with parameters: e.g., an interestingness measure threshold for the rules to be mined (sometimes coupled with a frequency threshold), the number of mined rules to use for building the final rule set for classification, etc. The performance of CBA-like classifiers strongly depends on parameter tuning. The choice of parameters is thus crucial but not trivial – each data set may require its own parameter settings. If tuning one parameter could be difficult, a common end-user could be quickly drowned into the tuning of several parameters.

These drawbacks suggest *(i)*, processing quantitative and categorical attributes directly (on the fly) in the mining process in order to catch multivariate correlations that are unreachable with univariate preprocessing and *(ii)* designing an interestingness measure with no need for any wise threshold tuning and a parameter-free method.

Contributions & Organization. In this paper, we suggest a new quantitative classification rule mining framework founded on a Bayesian approach. Our method draws its inspiration from the MODL approach (Minimum Optimized Description Length [5]) whose main concepts are recalled in the next section. In section 2, we instantiate the generic MODL approach for the case of classification rules; then, step by step, we build a Bayesian criterion which plays the role of an interestingness measure (with no need for thresholding) for classification rules and we discuss some of its fair properties. In section 3, we also suggest a new efficient parameter-free mining algorithm for the extraction of locally optimal classification rules. A classifier is then built following a simple and intuitive feature construction process based on the mined rules. The resulting classifier shows competitive results when compared with state-of-the-art rule-based classifiers on both real-world and large-scale challenge data sets – showing the added-value of the method (see section 4). Further related work is discussed in section 5 before concluding.

2 Towards MODL Rules

MODL Principle. The MODL approach is based on a Bayesian approach. Let us consider the univariate supervised discretization task as an example. From the MODL point of view, the problem of discretizing a numerical attribute is formulated as a model selection problem.

Firstly, a space $\mathcal{M}$ of discretization models M is defined. In order to choose the "best" discretization model M , a Bayesian "Maximum A Posteriori" approach (MAP) is used: the probability $p(M|D)$ is to be maximized over $\mathcal{M}$ (i.e., the posterior probability of a given discretization model M given the data D). Using Bayes rule and considering that $p(D)$ is constant in the current optimization problem, it consists in maximizing the expression $p(M) \times p(D|M)$. The prior $p(M)$ and the conditional probability $p(D|M)$ called the likelihood are both computed with the parameters of a specific discretization which is uniquely identified by the number of intervals, the bound of the intervals and the class frequencies in each interval. Notice that the prior exploits the hierarchy of parameters and is uniform at each stage of the hierarchy. The evaluation criterion is based on the negative logarithm of $p(M \mid D)$ and is called the cost of the model M: $c(M) = -\log(p(M) \times p(D \mid M))$. The optimal model M is then the one with the least cost c (see original work [5] for explicit expression of $p(M)$ and $p(D|M)$ and for the optimization algorithm). The generic MODL approach has also already been successfully applied to supervised value grouping [4] and decision tree construction [28]. In each instantiation, the MODL method promotes a trade-off between (1) the fineness of the predictive information provided by the model and (2) the robustness in order to obtain a good generalization of the model. Next, MODL approach is instantiated for the case of classification rules.

Basic Notations and Definitions. Let $r = \{\mathcal{T}, \mathcal{I}, \mathcal{C}\}$ be a labeled transactional data set, where $\mathcal{T} = \{t_1, \ldots, t_N\}$ is the set of objects, $\mathcal{I} = \{x_1, \ldots, x_m\}$ is a set of attributes (numerical or categorical) and $dom(x_j)$ the domain of an attribute x_j $(1 \leq j \leq m)$ and $\mathcal{C} = \{c_1, \ldots, c_J\}$ is the set of J mutually exclusive classes of a class attribute y. An object t is a vector $t = \langle v_1, \ldots, v_m, c\rangle$ where $v_j \in dom(x_j)$ $(1 \leq j \leq m)$ and $c \in \mathcal{C}$. An item for a numerical attribute x is an interval of the form $x[l_x, u_x]$ where $l_x, u_x \in dom(x)$ and $l_x \leq u_x$. We say that an object $t \in \mathcal{T}$ satisfies an interval item $x[l_x, u_x]$ when $l_x \leq t(x) \leq u_x$. For a categorical attribute, an item is a value group of the form $x\{v_x^1, \ldots, v_x^s\}$ where $v_x^j \in dom(x)$ $(1 \leq j \leq s)$. We say that an object $t \in \mathcal{T}$ satisfies a value group item $x\{v_x^1, \ldots, v_x^s\}$ when $t(x) \in \{v_x^1, \ldots, v_x^s\}$. An itemset X is just a set of items and an object t supports X if t satisfies all items of X. A classification rule π on r is an expression of the form $\pi : X \to c$ where c is a class value and X is an itemset. Notice that a categorical attribute involved in the rule body is partitioned into two value groups: the body item (or group) and the outside item; whereas a numerical attribute, due to the intrinsic order of its values, is discretized into either two or three intervals: the body item (or interval) and the outside item(s) (see example below).

Let us recall that, from a MODL point of view, the problem of mining a classification rule π is formulated as a model selection problem. To choose the best rule from the rule space we use a Bayesian approach: we look for maximizing $p(\pi|r)$. As explained in previous section, it consists in minimizing the cost of the rule defined as:

$$c(\pi) = -\log(p(\pi) \times p(r \mid \pi))$$

In order to compute the prior $p(\pi)$, we suggest another definition of classification rule based on hierarchy of parameters that uniquely identifies a given rule:

Definition 1 (Standard Classification Rule Model). *A* MODL *rule π, also called* standard classification rule model *(*SCRM*), is defined by:*

- *the constituent attributes of the rule body*
- *the group involved in the rule body, for each categorical attribute of the rule body*
- *the interval involved in the rule body, for each numerical attribute of the rule body*
- *the class distribution inside and outside of the body*

Notice that SCRM definition slightly differs from classical classification rule. The last key point meets the concept of distribution rule [17]. The consequent of a SCRM is an empirical distribution over the classes as illustrated in the following example:

Example of a SCRM. Let us consider the SCRM $\pi : (x_1 \in \{v^1_{x_1}, v^3_{x_1}, v^4_{x_1}\}) \wedge (1.2 < x_2 \leq 3.1) \wedge (x_4 \geq 100) \to (p_{c_1} = 0.9, p_{c_2} = 0.1)$ where x_1 is a categorical attribute and x_2, x_4 are numerical attributes. The value group $\{v^1_{x_1}, v^3_{x_1}, v^4_{x_1}\}$ and the intervals $]1.2; 3.1]$ and $[100; +\infty[$ are those items involved *in* the rule body. The complementary part (i.e. the negation of their conjunction) constitutes the *outside* part of the rule body. $(p_{c_1} = 0.9, p_{c_2} = 0.1)$ is the empirical class distribution for the objects covered by the rule body (inside part) and the class distribution for the outside part of the body may be deduced easily.

Our working model space is thus the space of all SCRM rules. To apply the Bayesian approach, we first need to define a prior distribution on the SCRM space; and we will need the following notations.

Notations. Let r be a data set with N objects, m attributes (categorical or numerical) and J classes. For a SCRM, $\pi : X \to (P_{c_1}, P_{c_2}, \ldots, P_{c_J})$ such that $|X| = k \leq m$, we will use the following notations:

- $X = \{x_1, \ldots, x_k\}$: the set of k constituent attributes of the rule body $(k \leq m)$
- $X_{cat} \cup X_{num} = X$: the sets of categorical and numerical attributes of the rule body
- $V_x = |dom(x)|$: the number of values of a categorical attribute x
- I_x: the number of intervals (resp. groups) of a numerical (resp. categorical) attribute x
- $\{i(v_x)\}_{v_x \in dom(x)}$: the indexes of groups to which v_x are affected (one index per value, either 1 or 2 for inside or outside of the rule body)
- $\{N_{i(x).}\}_{1 \leq i \leq I_x}$: the number of objects in interval i of numerical attribute x
- $i_{x_1}, \ldots, i_{x_k}$: the indexes of groups of categorical attributes (or intervals of numerical attributes) involved in the rule body
- $N_X = N_{i_{x_1} \ldots i_{x_k}}$: the number of objects in the body $i_{x_1} \ldots i_{x_k}$
- $N_{\neg X} = N_{\neg i_{x_1} \ldots i_{x_k}}$: the number of objects outside of the body $i_{x_1} \ldots i_{x_k}$

– $N_{Xj} = N_{i_{x_1} \dots i_{x_k} j}$: the number of objects of class j in the body $i_{x_1} \dots i_{x_k}$
– $N_{\neg Xj} = N_{\neg i_{x_1} \dots i_{x_k} j}$: the number of objects of class j outside of the body $i_{x_1} \dots i_{x_k}$

MODL Hierarchical Prior. We use the following distribution prior on SCRM models, called the MODL hierarchical prior. Notice that a uniform distribution is used at each stage[1] of the parameters hierarchy of the SCRM models:

- *(i)* the number of attributes in the rule body is uniformly distributed between 0 and m
- *(ii)* for a given number k of attributes, every set of k constituent attributes of the rule body is equiprobable (given a drawing with replacement)
- *(iii)* for a given categorical attribute in the body, the number of groups is necessarily 2
- *(iv)* for a given numerical attribute in the body, the number of intervals is either 2 or 3 (with equiprobability)
- *(v)* for a given categorical (or numerical) attribute, for a given number of groups (or intervals), every partition of the attribute into groups (or intervals) is equiprobable
- *(vi)* for a given categorical attribute, for a value group of this attribute, belonging to the body or not are equiprobable
- *(vii)* for a given numerical attribute with 2 intervals, for an interval of this attribute, belonging to the body or not are equiprobable. When there are 3 intervals, the body interval is necessarily the middle one.
- *(viii)* every distribution of the class values is equiprobable, in and outside of the body
- *(ix)* the distributions of class values in and outside of the body are independent

Thanks to the definition of the model space and its prior distribution, we can now express the prior probabilities of the model and the probability of the data given the model (i.e., $p(\pi)$ and $p(r \mid \pi)$).

Prior Probability. The prior probability $p(\pi)$ of the rule model is:

$$\begin{aligned} p(\pi) &= p(X) \\ &\times \prod_{x \in X_{cat}} p(I_x) \times p(\{i(v_x)\}|I_x) \times p(i_x|\{i(v_x)\}, I_x) \\ &\times \prod_{x \in X_{num}} p(I_x) \times p(\{N_{i(x).}\}|I_x) \times p(i_x|\{N_{i(x).}\}, I_x) \\ &\times p(\{N_{Xj}\}\{N_{\neg Xj}\} \mid N_X, N_{\neg X}) \end{aligned}$$

Firstly, we consider $p(X)$ (the probability of having the attributes of X in the rule body). The first hypothesis of the hierarchical prior is the uniform distribution of the number of constituent attributes between 0 and m. Furthermore, the second hypothesis says that every set of k constituent attributes of the rule body is equiprobable. The number of combinations $\binom{m}{k}$ could be a natural way to compute this prior; however, it is symmetric. Beyond $m/2$, adding new attributes makes the selection more probable. Thus, adding irrelevant variables is favored,

[1] It does not mean that the hierarchical prior is a uniform prior over the rule space, which would be equivalent to a maximum likelihood approach.

provided that this has an insignificant impact on the likelihood of the model. As we prefer simpler models, we suggest to use the number of combinations with replacement $\binom{m+k-1}{k}$. Using the two first hypothesis, we have:

$$p(X) = \frac{1}{m+1} \cdot \frac{1}{\binom{m+k-1}{k}}$$

For each categorical attribute x, the number of partitions of V_x values into 2 groups is $\mathcal{S}(V_x, 2)$ (where $\mathcal{S}$ stands for Stirling number of the second kind). Considering hypotheses *(iii)*, *(v)*, *(vi)*, we have:

$$p(I_x) = 1 \quad ; \quad p(\{i(v_x)\}|I_x) = \frac{1}{\mathcal{S}(V_x, 2)} \quad ; \quad p(i_x|\{i(v_x)\}, I_x) = \frac{1}{2}$$

For each numerical attribute x, the number of intervals is either 2 or 3. Computing the number of partitions of the (ranked) values into intervals turns into a combinatorial problem. Notice that, when $I_x = 3$ the interval involved in the rule body is necessarily the second one; when $I_x = 2$, it is either the first or the second with equiprobability. Considering hypotheses *(iv)*, *(v)*, *(vii)*, we get:

$$p(I_x) = \frac{1}{2} \quad ; \quad p(\{N_{i.}\}|I_x) = \frac{1}{\binom{N-1}{I_x-1}} \quad ; \quad p(i_x|\{N_{i.}\}, I_x) = \frac{1}{1 + \mathbb{1}_{\{2\}}(I_x)}$$

where $\mathbb{1}_{\{2\}}$ is the indicator function of set $\{2\}$ such that $\mathbb{1}_{\{2\}}(a) = 1$ if $a = 2$, 0 otherwise.

Using the hypotheses *(viii)* and *(ix)*, computing the probabilities of distributions of the J classes inside and outside of the rule body turns into a multinomial problem. Therefore, we have:

$$p(\{N_{Xj}\} \mid N_X, N_{\neg X}) = \frac{1}{\binom{N_X+J-1}{J-1}} \quad ; \quad p(\{N_{\neg Xj}\} \mid N_X, N_{\neg X}) = \frac{1}{\binom{N_{\neg X}+J-1}{J-1}}$$

The Likelihood. Now, focusing on the likelihood term $p(r \mid \pi)$, the probability of the data given the rule model is the probability of observing the data inside and outside of the rule body (w.r.t. to N_X and $N_{\neg X}$ objects) given the multinomial distribution defined for N_X and $N_{\neg X}$. Thus, we have:

$$p(r \mid \pi) = \frac{1}{\frac{N_X!}{\prod_{j=1}^{J} N_{Xj}!}} \cdot \frac{1}{\frac{N_{\neg X}!}{\prod_{j=1}^{J} N_{\neg Xj}!}}$$

Cost of a SCRM. We now have a complete and exact definition of the cost of a SCRM π:

$$
\begin{aligned}
c(\pi) = {} & \log(m+1) + \log \binom{m+k-1}{k} && (1)\\
& + \sum_{x \in X_{cat}} \log \mathcal{S}(V_x, 2) + \log 2 && (2)\\
& + \sum_{x \in X_{num}} \log 2 + \log \binom{N-1}{I_x - 1} + \log(1 + \mathbb{1}_{\{2\}}(I_x)) && (3)\\
& + \log \binom{N_X + J - 1}{J-1} + \log \binom{N_{\neg X} + J - 1}{J-1} && (4)\\
& + \left(\log N_X! - \sum_{j=1}^{J} \log N_{Xj}! \right) + \left(\log N_{\neg X}! - \sum_{j=1}^{J} \log N_{\neg Xj}! \right) && (5)
\end{aligned}
$$

The cost of a SCRM is the negative logarithm of probabilities which is no other than a coding length according to Shannon [25]. Here, $c(\pi)$ may be interpreted as the ability of a SCRM π to encode the classes given the attributes. Line (1) stands for the choice of the number of attributes and the attributes involved in the rule body. Line (2) is related to the choice of the groups and the values involved in the rule body for categorical attributes; line (3) is for the choice of the number of intervals, their bounds and the one involved in the rule body for numerical attributes. Line (4) corresponds to the class distribution in and outside of the rule body. Finally, line (5) stands for the likelihood.

Since the magnitude of the cost depends on the size of the data set (N and m), we defined a normalized criterion, called *level* and which plays the role of interestingness measure to compare two SCRM.

Definition 2 (Level: interestingness of SCRM). *The level of a* SCRM *is:*

$$
level(\pi) = 1 - \frac{c(\pi)}{c(\pi_\emptyset)}
$$

where $c(\pi_\emptyset)$ is the cost of the null model (i.e. default rule with empty body). Intuitively, $c(\pi_\emptyset)$ is the coding length of the classes when no predictive information is used from the attributes. The cost of the default rule $\pi_\emptyset$ is formally:

$$
c(\pi_\emptyset) = \log(m+1) + \log \binom{N+J-1}{J-1} + \left(\log N! - \sum_{j=1}^{J} \log N_j! \right)
$$

The *level* naturally draws the frontier between the interesting patterns and the irrelevant ones. Indeed, rules π such that $level(\pi) \leq 0$, are not more probable than the default rule $\pi_\emptyset$; then using them to explain the data is more costly than using $\pi_\emptyset$ – such rules are considered irrelevant. Rules such that $0 < level(\pi) \leq 1$ highlight the interesting patterns π. Indeed, rules with lowest cost (highest *level*) are the most probable and show correlations between the rule body and the class attribute. In terms of coding length, the *level* may also be interpreted as a compression rate. Notice also that $c(\pi)$ is smaller for lower k values (cf. line (1)), i.e. rules with shorter bodies are more probable thus preferable – which meets the consensus: "Simpler models are more probable and preferable". This idea is translated in the following proposition (the proof is almost direct):

Proposition 1. *Given two* SCRM π *and* π' *resp. with bodies* X *and* X'*, such that* $X \subseteq X'$ *and sharing the same contingency table (i.e.,* $N_X = N'_X$*,* $N_{\neg X} = N_{\neg X'}$*,* $N_{Xj} = N_{X'j}$*,* $N_{\neg Xj} = N_{\neg X'j}$*), then we have:* $c(\pi) < c(\pi')$ *and* π *is preferable.*

Asymptotic Behavior. The predominant term of the cost function is the likelihood term (eq.(5)) that indicates how accurate the model is. The others terms behave as regularization terms, penalizing complex models (e.g., with too many attributes involved in the rule body) and preventing from over-fitting. The following two theorems show that the regularization terms are negligible when the number of objects N of the problem is very high and that the cost function is linked with Shannon class-entropy [10] (due to space limitations, full proofs are not given, but the key relies on the Stirling approximation: $\log n! = n(\log n - 1) + O(\log n)$).

Theorem 1. *The cost of the default rule* $\pi_\emptyset$ *for a data set made of* N *objects is asymptotically* N *times the Shannon class-entropy of the whole data set when* $N \to \infty$*, i.e.* $H(y) = -\sum_{j=1}^{J} p(c_j) \log p(c_j)$*.*

$$\lim_{N\to\infty} \frac{c(\pi_\emptyset)}{N} = -\sum_{j=1}^{J} \frac{N_j}{N} \log \frac{N_j}{N}$$

Theorem 2. *The cost of a rule* π *for a data set made of* N *objects is asymptotically* N *times the Shannon conditional class-entropy when* $N \to \infty$*, i.e.* $H(y|x) = -\sum_{x\in\{X,\neg X\}} p(x) \sum_{j=1}^{J} p(c_j|x) \log p(c_j|x)$*.*

$$\lim_{N\to\infty} \frac{c(\pi)}{N} = \frac{N_X}{N}\left(\sum_{j=1}^{J} -\frac{N_{Xj}}{N_X}\log\frac{N_{Xj}}{N_X}\right) + \frac{N_{\neg X}}{N}\left(\sum_{j=1}^{J} -\frac{N_{\neg Xj}}{N_{\neg X}}\log\frac{N_{\neg Xj}}{N_{\neg X}}\right)$$

The asymptotic equivalence between the coding length of the default rule $\pi_\emptyset$ and the class-entropy of the data confirms that "rules such that *level* ≤ 0 identify patterns that are not statistically significant" and links the MODL approach with the notion of incompressibility of Kolmogorov [21] – which defines randomness as the impossibility of compressing the data shorter than its raw form.

The asymptotic behavior of the cost function (for a given rule π) confirms that high *level* values highlight the most probable rules that characterize classes, since high *level* value means high class-entropy ratio between π and the default rule. In terms of compression, rules with *level* > 0 correspond to a coding with better compression rate than the default rule; thus, they identify patterns that do not arise from randomness. Here, we meet the adversarial notions of spurious and significant patterns as mentioned and studied in [31]. Conjecture 1 illustrates this idea and we bring some empirical proof to support it in Section 4:

Conjecture 1. Given a classification problem, for a random distribution of the class values, there exist no SCRM with *level* > 0 (asymptotically according to N, almost surely).

Problem Formulation. Given the MODL method framework instantiated for classification rules, an ambitious problem formulation would have been: "*Mining the whole set of* SCRM *with level* > 0" (or the set of K-top *level* SCRM). However, the model space is huge, considering all possibilities of combinations of attributes, attribute discretization and value grouping: the complexity of the problem is $O((2^{V_c})^{m_c}(N^2)^{m_n})$ where m_c is the number of categorical attributes with V_c values and m_n the number of numerical attributes. Contrary to some standard approaches for classification rule mining, exhaustive extraction is not an option. Our objective is to sample the posterior distribution of rules using a randomization strategy, starting from rules (randomly) initialized according to their prior. Therefore, we opt for a simpler formulation of the problem: "*Mining a set of locally optimal* SCRM *with level* > 0". In the following, we describe our mining algorithm and its sub-routines for answering the problem.

3 MODL Rule Mining

This section describes our strategy and algorithm for mining a set of locally optimal SCRM (algorithm 1 MACATIA[2]) and how we use it in a classification system.

Algorithm 1. MACATIA: The MODL-rule miner

```
   Input  : $r = \{\mathcal{T}, \mathcal{I}, \mathcal{C}\}$ a labeled data set
   Output: $\Gamma$ a set of locally optimal SCRM
1  $\Gamma \leftarrow \emptyset$;
2  while ¬ StoppingCondition do
3      $t \leftarrow$ chooseRandomObject($\mathcal{T}$);
4      $X \leftarrow$ chooseRandomAttributes($\mathcal{I}$);
5      $\pi \leftarrow$ initRandomBodyRule($X$,$t$);
6      currentLevel ← computeRuleLevel($\pi$,$r$);
7      repeat
8          minLevel ← currentLevel;
9          randomizeOrder($X$);
10         for $x \in X$ do
11             OptimizeAttribute($t, x, X$);
12         deleteNonInformativeAttributes(X);
13         currentLevel ← computeRuleLevel($\pi$,$r$);
14     until noMoreLevelImprovement;
15     if currentLevel > 0 then
16         $\Gamma \leftarrow \Gamma \cup \{\pi\}$;
17 return $\Gamma$
```

The MODL Rule Miner. We adopt an instance-based randomized strategy for mining rules in given allowed time. The stopping condition (l.2) is the time that the end-user grants to the mining process. At each iteration of the main loop (l.2-16), a locally optimal SCRM is built – when time is up, the process ends and the current rule set is output. Firstly (l.3-5), a random object t and a random set of k attributes are chosen from the data set; then, a SCRM π is randomly initialized such that the body of π is made of a random itemset based on attributes X and t supports the rule body (to simplify notations, X

[2] MACATIA refers to a typical sweet bun from Reunion island.

and the body itemset based on X own the same notation). The inner loop (l.7-14) optimizes the current rule while preserving the constraint "t supports body itemset X". We are looking for *level* improvement: a loop of optimization consists in randomizing the order of the body attributes, optimizing each item (attribute) sequentially – the intervals or groups of an attribute are optimized while the other body attributes are fixed (see specific instantiations of OptimizeAttribute in sub-routines algorithms 2 and 3), then removing non-informative attributes from the rule body (i.e., attributes with only one interval or only one value group). Optimization phase ends when there is no more improvement. Finally, the optimized rule is added to the rule set if its *level* is positive.

Attribute Optimization. Let us remind that, while optimizing a rule, each rule attribute (item) is optimized sequentially while the others are fixed.

For a numerical attribute x (see algorithm 2), we are looking for the best bounds for its body interval containing $t(x)$ (i.e. the bounds that provide the best *level* value for the current SCRM while other attributes are fixed). If there are two intervals (l.3-4), only one bound is to be set and the best one is chosen among all the possible ones. When there are three intervals (l.1-2), the lower bound and the upper bound of the body interval are to be set. Each bound is set sequentially and in random order (again, the best one is chosen while the other is fixed). Since an interval might be empty at the end of this procedure, we remove empty intervals (l.5) – the current attribute might be deleted from the rule body by the main algorithm if only one interval is remaining.

Algorithm 2. OptimizeAttribute: Numerical attribute optimization

```
Input  : $r = \{\mathcal{T}, \mathcal{I}, \mathcal{C}\}$ a transactional labeled data set, $\pi : X \to (p_{c_J}, \dots, p_{c_J})$ a SCRM
         covering an object $t \in \mathcal{T}$, $x \in X$ a numerical attribute of the rule body
Output: $x$ an optimized numerical attribute
1 if $x$.IntervalsNumber == 3 then
2     ($x$.LB,$x$.UB) ← chooseBestBounds($t, x, \pi, r$);
3 if $x$.IntervalsNumber == 2 then
4     $x$.B ← chooseBestBound($t, x, \pi, r$);
5 cleanEmptyIntervals();
6 return $x$
```

Algorithm 3. OptimizeAttribute: Categorical attribute optimization

```
Input  : $r = \{\mathcal{T}, \mathcal{I}, \mathcal{C}\}$ a transactional labeled data set, $\pi : X \to (p_{c_J}, \dots, p_{c_J})$ a SCRM
         covering an object $t \in \mathcal{T}$, $x \in X$ a categorical attribute of the rule body
Output: $x$ an optimized categorical attribute
minLevel ← computeRuleLevel($\pi$,$r$);
currentLevel ← computeRuleLevel($\pi$,$r$);
Shuffle($x$.allValues);
for value ∈ {$x$.allValues \{$t(x)$}} do
    changeGroup(value);
    currentLevel ← computeRuleLevel($\pi$,$r$);
    if currentLevel > minLevel then
        minLevel ← currentLevel;
    else
        ChangeGroup(value);
cleanEmptyGroups();
return $x$
```

For a categorical attribute x (see algorithm 3), we are looking for a partition of the value set into two value groups (i.e. the value groups that provide the best *level* value for the current SCRM while other attributes are fixed). First (l.3), the values of the current categorical attribute are shuffled. Then (l.5-10), we try to transfer each value (except for $t(x)$ staying in the body group) from its origin group to the other: the transfer is performed if the *level* is improved. Once again we clean possible empty value group at the end of the procedure (necessarily the out-body group) – the current attribute might be deleted from the rule body by the main algorithm if only one group is remaining.

About Local Optimality. Our MODL rule miner (and its sub-routines) bet on a trade-off between optimality and efficiency. In the main algorithm, the strategy of optimizing an attribute while the other are fixed leads us to a local optimum, given the considered search neighborhood (so do the strategies of optimizing interval and value group items). This trade-off allows us to mine *one* rule in time complexity $O(kN \log N)$ using efficient implementation structures and algorithms. Due to space limitation, we cannot give details about the implementation.

About Mining with Diversity. Randomization is present at each stage of our algorithm. Notice that the randomization is processed according the defined and motivated hierarchical prior (except for object choice). As said above, we are not looking for an exhaustive extraction but we want to sample the posterior distribution of SCRM rules. This randomization facet of our method (plus the optimization phase) allows us mine interesting rules ($level > 0$) with diversity.

Classification System. We adopt a simple and intuitive feature construction process to build a classification system based on a set of locally optimal SCRM. For each mined rule π, a new Boolean attribute (feature) is created: the value of this new feature for a training object t of the data set r is (1) true if t supports the body of π, (0) false otherwise. This feature construction process is certainly the most straightforward but has also shown good predictive performance in several studies [8,14]. To provide predictions for new incoming (test) objects, we use a Selective Naive Bayes classifier (SNB) on the recoded data set. This choice is motivated by the good performances of SNB on benchmark data sets as well as on large-scale challenge data sets (see [6]). Moreover, SNB is Bayesian-based and parameter-free, agreeing with the characteristics of our method.

4 Experimental Validation

In this section, we present our empirical evaluation of the classification system (noted KRSNB). Our classification system has been developed in C++ and is using a JAVA-based user interface and existing libraries from KHIOPS[3]. The experiments are performed to discuss the following questions:

$\mathcal{Q}_1$ The main algorithm is controlled by a running time constraint. In a given allowed time, a certain number of rules might be mined: how do the

[3] http://www.khiops.com

performance of the classifier system evolve w.r.t. the number of rules? And, what about the time-efficiency of the process?

$\mathcal{Q}_2$ Does our method suffer over-fitting? What about spurious patterns? We will also bring an empirical validation of conjecture 1.

$\mathcal{Q}_3$ Does the new feature space (made of locally optimal rules) improve the predictive performance of SNB?

$\mathcal{Q}_4$ Are the performance of the classification system comparable with state-of-the-art CBA-like classifiers?

For our experiments, we use 29 UCI data sets commonly used in the literature (australian, breast, crx, german, glass, heart, hepatitis, horsecolic, hypothyroid ionosphere, iris, LED, LED17, letter, mushroom, pendigits, pima, satimage, segmentation, sickeuthyroid, sonar, spam, thyroid, tictactoe, vehicle, waveform and its noisy version, wine and yeast) and which show a wide variety in number of objects, attributes and classes, in the type of attributes and the class distribution (see [2] for a full description). All performance results reported in the following are obtained with stratified 10-fold cross validation. Notice that, the feature construction step is performed *only on the training set* and the new learned features are reported on the test set for each fold of the validation.

Evolution of Performance w.r.t. the Number of Rules. In figure 2, we plot the performance in terms of accuracy and AUC of KRSNB based on ρ rules ($\rho = 2^n, n \in [0, 10]$). The details per data set is not as important as the general behavior: we see that, generally, the predictive performance (accuracy and AUC) increases with the number of rules. Then, the performance reaches a plateau for most of the data sets: with about a few hundreds of rules for accuracy and a few rules for AUC.

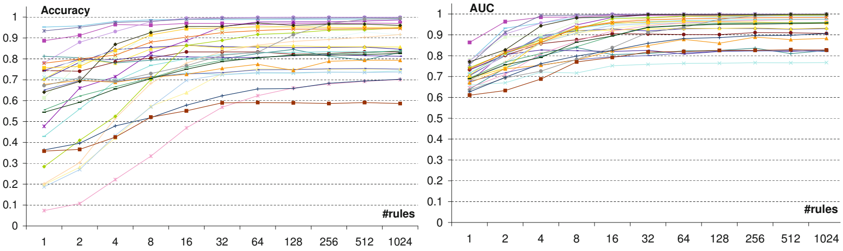


Fig. 2. Accuracy and AUC results (per data set) w.r.t. number of rules mined

Running Time Report. Due to our mining strategy, running time grows linearly with the number of rules to be mined. For most of the data sets, mining a thousand rules is managed in less than 500s. In fig. 3, for each of the 29 data sets, we report the processing time of KRSNB based on 1024 rules w.r.t. the size of the data set – plotted with logarithmic scales. It appears that the MODL-rule miner roughly runs in linear time according to $N \times m$.

The analysis of performance evolution and running time w.r.t. the number of rules shows that KRSNB reaches its top performance in reasonable time using a few hundreds of rules. In the following, to facilitate the presentation, we will experiment our classifier with 512 rules.

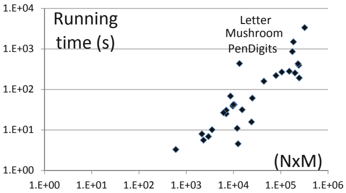


Fig. 3. Running time for mining 1024 rules w.r.t. the size of the data set ($N \times m$)

About Spurious Patterns and Robustness of Our Method. As mentioned in [31], "*Empirical studies demonstrate that standard pattern discovery techniques can discover numerous spurious patterns when applied to random data and when applied to real-world data result in large numbers of patterns that are rejected when subjected to sound statistical validation*". Proposition 1 states that in a data set with random class distribution, there should not exist any SCRM with $level > 0$ (i.e. no interesting rule). To support this proposition, we lead the following empirical study: *(i)* for each of the 29 benchmark data sets, we randomly assign a class label $c \in \mathcal{C}$ to the objects; *(ii)* we run KRSNB on the data sets with random labels. The result is strong: all rule optimizations during the process end with a default rule with $level \leq 0$. This study shows that our method is robust, discovers no spurious patterns and thus avoids over-fitting.

Added-Value of the New Feature Space. We process a comparative study of the performance of SNB and KRSNB to demonstrate the added-value of the new feature space. Due to space limitations, we skip results on individual data sets and only Area Under ROC Curve (AUC) and accuracy average results of each method are reported in table 1. We are aware of the problems arising from averaging results over various data sets, therefore Win-Tie-Loss (WTL) and average rank results are also reported. A raw analysis of the results gives advantage to KRSNB (in all dimensions: average accuracy and AUC, rank and WTL results). Concerning the Win-Tie-Loss results (WTL) at significance level $\alpha = 0.05$, the critical value for the two-tailed sign test is 20 (for 29 data sets). Thus, even if we cannot assert a significant difference of AUC performance between the two approaches, WTL AUC results of KRSNB vs SNB is close to the critical value ($17 < 20$) – which is a promising result. Moreover, the new feature space made of locally optimal SCRM is clearly a plus when considering WTL accuracy results.

Table 1. Comparison of SNB and KRSNB performance results

	Accuracy		AUC	
Algorithms	avg	rank	avg	rank
SNB	83.58	1.72	92.43	1.60
KRSNB	84.80	1.28	93.48	1.39
WTL KR VS. SNB	**21**/0/8		**17**/2/10	

Table 2. Comparison of KRSNB with state-of-the-art methods

Algorithms	avg.acc	avg.rank	KR-WTL
KRSNB	84.80	2.17	-
HARMONY	83.31	3.53	19/1/9
KRIMP	83.31	3.64	23/1/5
RIPPER	84.38	2.83	19/1/9
PART	84.19	2.83	18/1/10

Comparisons with State-of-the-Art. Let us first notice that, for the tiny XOR case shown in introduction, KRSNB easily finds the four obvious 2-dimensional patterns characterizing the classes – this finding is unreachable for CBA-like methods using univariate pre-processing. We now compare the performance of KRSNB with four state-of-the-art competitive rule-based classifiers: two recent pattern-based classifiers: HARMONY [29] an instance-based classifier and KRIMP [20] a compression-based method; and two induction-rule-based approaches RIPPER [9] and PART [12] available from the WEKA platform [16] with default parameters. The choice of accuracy for performance comparisons is mainly motivated by the fact that competitors (HARMONY and KRIMP) provide only accuracy results. Since HARMONY and KRIMP are restricted to Boolean (or categorical) data sets, we preprocess the data using a MDL-based univariate supervised discretization [16] for these methods.

We also run experiments with parameters as indicated in the original papers. Once again, only average results are reported in table 2. A first analysis of the raw results shows that KRSNB is highly competitive. Again, average accuracy, Win-Tie-Loss and average rank results give advantage to KRSNB. We also applied the Friedman test coupled with a post-hoc Nemenyi test as suggested by [11] for multiple comparisons (at significance level $\alpha = 0.05$ for both tests). The null-hypothesis was rejected, which indicates that the compared classifiers are not equivalent in terms of accuracy. The result of the Nemenyi test is represented by the critical difference chart shown in figure 4 with $CD \simeq 1.133$. First of all, we observe that there is no critical difference of performance between the four competitors of KRSNB. Secondly, even if KRSNB is not statistically singled out, it gets a significant advantage on HARMONY and KRIMP – whereas PART and RIPPER do not get this advantage.

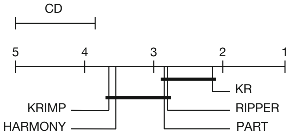


Fig. 4. Critical difference of performance between KRSNB and state-of-the-art rule-based classifiers

Results on Challenge Data Sets. We also experiment KRSNB on recent large-scale challenge data sets (Neurotech challenges at PAKDD'09-10 and Orange *small* challenge at KDD'09)[4]. Each data set involves 50K instances, from tens to hundreds quantitative attributes, and two highly imbalanced classes – recognized as a difficult task. We experiment KRSNB and its competitors in a 70%train-30%test setting and report AUC results in table 3. As univariate preprocessing of quantitative attributes generate thousands of variables, we were not able to obtain any results with KRIMP and HARMONY. Thus, a first victory for KRSNB is its ability to mine rules from large-scale data. Secondly, it appears that the class-imbalance facet of the tasks severely harms the predictive performance of RIPPER and PART; there, KRSNB outperforms its competitors.

Table 3. Comparison of AUC results for challenge data sets

	NEUROTECH-PAKDD		ORANGE-KDD'09		
	2009	2010	APPET.	CHURN	UPSELL.
KRSNB	66.31	62.27	82.02	70.59	86.46
RIPPER	51.90	50.70	50.00	50.00	71.80
PART	59.40	59.20	76.40	64.70	83.50

[4] http://sede.neurotech.com.br/PAKDD2009/;
http://sede.neurotech.com.br/PAKDD2010/ ; http://www.kddcup-orange.com/

5 Discussion and Related Work

As mentioned in sec. 2, the MODL method and its extension to classification rules are at the crossroads of Bayes theory and Kolmogorov complexity [21]. Our approach is also related to Minimum Description Length principle (MDL [15]) since the cost of a rule is similar to a coding length.

About MDL, Information Theory and Related. Some traditional rule learning methods integrate MDL principle in their mining algorithms *(i)* as a stopping criterion when growing a rule and/or *(ii)* as a selection criterion for choosing the final rule set (see e.g. [9,23]).

The MODL method is similar to practical MDL (also called crude MDL) which aims at coding the parameters of models M and data D given the models by minimizing the total coding length $l(M) + l(D|M)$. In [20], authors develop a MDL-based pattern mining approach (KRIMP) and its extension for classification purpose. The main divergences with our work are: *(i)* the MODL hierarchical prior induces a different way of coding information; *(ii)* KRIMP is designed for Boolean data sets and works with parameters when MODL is parameter-free and handles quantitative data sets. Also related to information theory, based on recently introduced maximum entropy models, [19] suggest the ratio of Shannon information content over the description length of a tile (i.e. an itemset coupled with its support) as an interestingness measure for binary tiles in an exploratory framework.

About Mining Quantitative Data Sets. The need for handling quantitative attributes in pattern mining tasks is not new. Srikant & Agrawal [26] develop a method for mining association rule in quantitative data sets: they start from a fine partitioning of the values of quantitative attributes, then combine adjacent partitions when interesting. After pioneering work, the literature became abundant; see e.g., [30,18]. The main differences with our work come from *(i)* the way of dealing with numerical attributes *(ii)* the mining strategy. Many methods start from a fine-granularity partition of the values and then try to merge or combine them – we design on-the-fly optimized intervals and groups when mining rules. Moreover, they inherit from classical association rule framework in which parameters are to be set.

About Mining Strategy and Sampling Methods. Exhaustive search might be inefficient on large-scale binary data (with many attributes). When facing quantitative attributes, the task is much more complicated. Separate-and-conquer (or covering) strategies [13] greedily extend one rule at once and follows a sequential data coverage scheme to produce the rule set; these strategies can tackle with large data with quantitative attributes. Our randomized strategy promotes diversity by sampling the posterior distribution of SCRMs. However, we are aware of very recent pattern mining algorithms for *binary* data using advanced sampling methods like Markov chains Monte Carlo methods (see e.g. [3,27]). Notice that our method, coupling randomized sampling with instance-based strategy, may generate similar rules. As SNB is quasi-insensitive to redundant features [6], it does not echo in the predictive performance of the classification system. We did not focus on the redundancy and sampling issues in this first study, but they are planned for future work.

6 Conclusion

We have suggested a novel framework for classification rule mining in quantitative data sets. Our method stems from the generic MODL approach. The present instantiation has lead us to several significant contributions to the field: *(i)* we have designed a new interestingness measure (*level*) that allows us to naturally mark out interesting and robust classification rules; *(ii)* we have developed a randomized algorithm that efficiently mines interesting and robust rules with diversity; *(iii)* the resulting classification system is parameter-free, deals with quantitative attributes without pre-processing and demonstrates highly competitive inductive performance compared with state-of-the-art rule-based classifiers while being highly resilient to spurious patterns. The genericity of the MODL approach and its present successful instantiation to classification rules call for other intuitive extensions, e.g., for regression rules or for other pattern type in an exploratory framework (such as descriptive rule or sequence mining).

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: ACM SIGMOD 1993, pp. 207–216 (1993)
2. Asuncion, A., Newman, D.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml/
3. Boley, M., Gärtner, T., Grosskreutz, H.: Formal concept sampling for counting and threshold-free local pattern mining. In: SIAM DM 2010, pp. 177–188 (2010)
4. Boullé, M.: A bayes optimal approach for partitioning the values of categorical attributes. Journal of Machine Learning Research 6, 1431–1452 (2005)
5. Boullé, M.: MODL: A bayes optimal discretization method for continuous attributes. Machine Learning 65(1), 131–165 (2006)
6. Boullé, M.: Compression-based averaging of selective naive Bayes classifiers. Journal of Machine Learning Research 8, 1659–1685 (2007)
7. Bringmann, B., Nijssen, S., Zimmermann, A.: Pattern-based classification: A unifying perspective. In: LeGo 2009 Workshop @ EMCL/PKDD 2009 (2009)
8. Cheng, H., Yan, X., Han, J., Hsu, C.W.: Discriminative frequent pattern analysis for effective classification. In: Proceedings ICDE 2007, pp. 716–725 (2007)
9. Cohen, W.W.: Fast effective rule induction. In: ICML 1995, pp. 115–123 (1995)
10. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley (2006)
11. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
12. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: ICML 1998, pp. 144–151 (1998)
13. Fürnkranz, J.: Separate-and-conquer rule learning. Artificial Intelligence Revue 13(1), 3–54 (1999)

14. Gay, D., Selmaoui, N., Boulicaut, J.-F.: Feature Construction Based on Closedness Properties Is Not That Simple. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 112–123. Springer, Heidelberg (2008)
15. Grünwald, P.: The minimum description length principle. MIT Press (2007)
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Expl. 11(1), 10–18 (2009)
17. Jorge, A.M., Azevedo, P.J., Pereira, F.: Distribution Rules with Numeric Attributes of Interest. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 247–258. Springer, Heidelberg (2006)
18. Ke, Y., Cheng, J., Ng, W.: Correlated pattern mining in quantitative databases. ACM Transactions on Database Systems 33(3) (2008)
19. Kontonasios, K.N., de Bie, T.: An information-theoretic approach to finding informative noisy tiles in binary databases. In: SIAM DM 2010, pp. 153–164 (2010)
20. van Leeuwen, M., Vreeken, J., Siebes, A.: Compression Picks Item Sets That Matter. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 585–592. Springer, Heidelberg (2006)
21. Li, M., Vitányi, P.M.B.: An Introduction to Kolmogorov Complexity and Its Applications. Springer (2008)
22. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings KDD 1998, pp. 80–86 (1998)
23. Pfahringer, B.: A New MDL Measure for Robust Rule Induction. In: Lavrač, N., Wrobel, S. (eds.) ECML 1995. LNCS, vol. 912, pp. 331–334. Springer, Heidelberg (1995)
24. Quinlan, J.R., Cameron-Jones, R.M.: FOIL: A Midterm Report. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 3–20. Springer, Heidelberg (1993)
25. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal (1948)
26. Srikant, R., Agrawal, R.: Mining quantitative association rules in large relational tables. In: SIGMOD 1996, pp. 1–12 (1996)
27. Tatti, N.: Probably the best itemsets. In: KDD 2010, pp. 293–302 (2010)
28. Voisine, N., Boullé, M., Hue, C.: A bayes evaluation criterion for decision trees. In: Advances in Knowledge Discovery & Management, pp. 21–38. Springer (2010)
29. Wang, J., Karypis, G.: HARMONY : efficiently mining the best rules for classification. In: Proceedings SIAM DM 2005, pp. 34–43 (2005)
30. Webb, G.I.: Discovering associations with numeric variables. In: KDD 2001, pp. 383–388 (2001)
31. Webb, G.I.: Discovering significant patterns. Machine Learning 68(1), 1–33 (2007)
32. Yin, X., Han, J.: CPAR : Classification based on predictive association rules. In: Proceedings SIAM DM 2003, pp. 369–376 (2003)

A Bayesian Scoring Technique for Mining Predictive and Non-Spurious Rules

Iyad Batal[1], Gregory Cooper[2], and Milos Hauskrecht[1]

[1] Department of Computer Science, University of Pittsburgh
[2] Department of Biomedical Informatics, University of Pittsburgh

Abstract. Rule mining is an important class of data mining methods for discovering interesting patterns in data. The success of a rule mining method heavily depends on the evaluation function that is used to assess the quality of the rules. In this work, we propose a new rule evaluation score - the Predictive and Non-Spurious Rules (PNSR) score. This score relies on Bayesian inference to evaluate the quality of the rules and considers the structure of the rules to filter out spurious rules. We present an efficient algorithm for finding rules with high PNSR scores. The experiments demonstrate that our method is able to cover and explain the data with a much smaller rule set than existing methods.

1 Introduction

The large amounts of data collected today provide us with an opportunity to better understand the behavior and structure of many natural and man-made systems. Rule mining is an important direction of machine learning and data mining research, which aims to elicit knowledge from data in terms of if-then rules that are intuitive and easy to understand by humans.

In this work, we study and apply rule mining to discover patterns in supervised learning tasks, where we have a specific target variable (outcome) and we want to find patterns (subpopulations of data instances) where the distribution of the target variable is statistically "most interesting". Examples of such patterns are: "subpopulation of patients who smoke and have a positive family history are at a significantly higher risk for lung cancer than the rest of the patients". This task has a high practical relevance in many domains of science or business. For example, finding a pattern that clearly and concisely defines a subpopulation of patients that respond better (or worse) to a certain treatment than the rest of the patients can speed up the validation process of this finding and its future utilization in patient-management.

In order to perform supervised rule discovery, we need to define a *search algorithm* to explore the space of potential rules and a *scoring function* to assess the interestingness of the rules. In this work, we use Frequent Pattern Mining (FPM) [1] to search for rules. The advantage of FPM is that it performs a more systematic search than heuristic rule induction approaches, such as greedy sequential covering [7–9]. However, its main disadvantage is that it often produces

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 260–276, 2012.

a large number of rules. Moreover, many of these rules are spurious because they can be naturally explained by other simpler (more general) rules. Therefore, it is crucial to devise an effective scoring function that allows us to select important and non-redundant rules from a large pool of frequent patterns.

To achieve this goal, we introduce the *Predictive and Non-Spurious Rules* (*PNSR*) score. This score applies Bayesian inference to evaluate the quality of individual rules. In addition, it considers the structure of patterns to assure that every rule is not only predictive with respect to the general population, but also with respect to all of its simplifications (generalizations). We show that using our score to mine the top rules, we are able to cover and explain the data with much fewer rules compared with classical supervised rule discovery methods. Finally, we present an efficient algorithm that integrates rule evaluation with frequent pattern mining and applies pruning strategies to speed up the mining.

2 Supervised Descriptive Rule Discovery

In this work, we are interested in applying rule mining in the *supervised setting*, where we have a special variable of interest Y (the target variable) and we want to mine rules that can help us to uncover "interesting" dependencies between Y and the input variables (attributes).

The dominant paradigm for supervised rule induction is to apply a sequential covering method [7–9], which learns a set of rules by first learning a single rule, removing the positive instances it covers and then repeating the process over the remaining instances. However, this approach is not appropriate for knowledge discovery because the rules are induced from biased data (including only positive instances not covered by previous rules). Therefore, the rules are difficult to interpret and understand by the user.

In contrast to the sequential covering approach, our task is to find a set of *comprehensible* rules/patterns that are statistically interesting with respect to the entire data, e.g., the rules should have wide coverage and unusual distributional characteristics with respect to the target variable [18]. This task appeared in the literature under a variety of different names, such as contrast set mining [2], emerging pattern mining [11] and subgroup discovery [17, 18]. Later on, [23] provided a unifying framework of this work which is named *Supervised Descriptive Rule Discovery* (*SDRD*).

To apply *SDRD*, we need to define a search algorithm to explore the space of potential rules and a scoring function S (quality measure) to assess the interestingness of each rule (S maps each rule R_i to a real number $S(R_i) \in \mathbb{R}$ that reflects its importance). Our objective in this work is to design a function S such that the top rules do not only predict well the target class variable compared to the entire population, but are also non-spurious in that their prediction is better than all of their generalizations (simplifications).

2.1 Definitions

Let $D = \{x_i, y_i\}_{i=1}^{n}$ be our data, where each instance x_i is described by a fixed number of attributes and is associated with a class label $y_i \in dom(Y)$. We assume that all attributes have discrete values (numeric attributes must be discretized [13, 28]).

We call every attribute-value pair an *item* and a conjunction of items a *pattern*. A pattern that contains k items is called a *k-pattern*. For example, $Education = PhD \wedge Marital\text{-}status = Single$ is a *2-pattern*.

Pattern P is a *subpattern* of pattern P', denoted as $P \subset P'$, if every item in P is contained in P' and $P \neq P'$. In this case, P' is a *superpattern* of P. For example, P_1 :*Education=PhD* is a subpattern of P_2 :*Education=PhD* $\wedge$ *Marital-status=Single*. This subpattern (*more-general-than*) relation defines a *partial ordering* of patterns, i.e. a *lattice structure*, as shown in Figure 1.

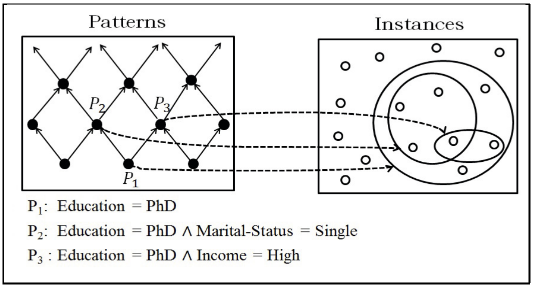

Patterns
Instances
P_2
P_3
P_1
P1: Education = PhD
P2: Education = PhD ∧ Marital-Status = Single
P3 : Education = PhD ∧ Income = High

Fig. 1. The box on the left shows the set of all patterns and the box on the right shows the set of all instances. Each pattern is associated with a group of instances that satisfy the pattern. The patterns are organized in a lattice structure according to the subpattern-superpattern relation.

Instance x_i satisfies pattern P, denoted as $P \in x_i$, if every item in P is present in x_i. Every pattern P defines a *group* (subpopulation) of the instances that satisfy P: $G_P = \{(x_i, y_i) : x_i \in D \wedge P \in x_i\}$. If we denote the empty pattern by ϕ, G_ϕ represents the entire data D. Note that $P \subset P'$ (P is a subpattern of P') implies that $G_P \supseteq G_{P'}$ (see Figure 1).

The *support* of pattern P in dataset D, denoted as $sup(P, D)$, is the number of instances in D that satisfy P (the size of G_P). Given a user defined *minimum support* threshold σ, P is called a *frequent pattern* if $sup(P, D) \geq \sigma$.

A rule is defined as $P \Rightarrow y$, where P (the condition) is a pattern and $y \in dom(Y)$ (the consequent) is a class label. We say that $P \Rightarrow y$ is a *subrule* of $P' \Rightarrow y'$ if $P \subset P'$ and $y = y'$. The *coverage* of rule $P \Rightarrow y$ is the proportion of instances in the data that satisfy P. The *confidence* of rule $P \Rightarrow y$, denoted as $conf(P \Rightarrow y)$, is the proportion of instances from class y among all the instances that satisfy P, i.e., it is the maximum likelihood estimation of $Pr(Y = y|P)$.

2.2 Rule Evaluation

A straightforward approach to *SDRD* is to use a rule quality measure (cf [14]) to score each rule by contrasting it to the general population (the entire data) and report the top rules to the user. We will argue that this approach is ineffective and can lead to many spurious (redundant) rules. We start by illustrating the spurious rules problem using an example and then describe it more formally in Section 2.3.

Example 1. Assume our objective is to identify populations of patients who are at high risk of developing coronary heart disease (CHD). Assume our dataset contains 150 instances, 50 of them are CHD cases and the others are controls. That is, the CHD prior, i.e. $conf(\Phi \Rightarrow CHD)$, is $50/150 \approx 33.3\%$.

Now, our task is to evaluate the following 3 rules:

- R_1:Race=White ⇒ CHD
  [#cases=29, #controls=61, conf=32.2%]
- R_2:Family history=Yes ⇒ CHD
  [#cases=30, #controls=20, conf=60%]
- R_3:Family history=Yes ∧ Race=White ⇒ CHD
  [#cases=21, #controls=11, conf=65.6%]

For each rule, we show the number of CHD cases and the number of controls that the rule covers. We also show the confidence of the rule.

One of the commonly used approaches to filter out uninteresting rules is to apply the χ^2 *test* to assure that there is a significant positive correlation between the condition and the consequent of each rule [2, 5, 20, 22]. If we apply the χ^2 test on our three rules, the p-values we get for R_1, R_2, and R_3 are 0.724, 9.6×10^{-7}, and 1.2×10^{-5}, respectively. That is, both R_2 and R_3 are statistically (very) significant with respect to a significance level $\alpha = 0.05$. Moreover, these two rules will be considered interesting using most rule quality measures [14].

[3] proposed the *confidence improvement* constraint, which says that each rule in the result should have a higher confidence than all of its subrules:

$$conf(P \Rightarrow y) - \max_{S \subset P}\{conf(S \Rightarrow y)\} > 0$$

This filter have been used quite a lot in the rule mining literature [15, 19, 20, 26]. If we applied the confidence improvement constraint to our working example, both R_2 and R_3 will be retained.

As we can see, both χ^2 test and confidence improvement agree that R_3 is an interesting rule. However, this rule may seem predictive only because it contains a simpler predictive rule (R_2). So should we consider R_3 to be interesting (show it to the user) or spurious? We will revisit this question after introducing the PNSR score.

2.3 Spurious Rules

Spurious rules are formed by adding irrelevant items to the antecedent of a simpler predictive rule. Let us illustrate this using the simple Bayesian belief network in Figure 2. In this network, the value of the class variable Y only depends on the value of feature F_1 and is independent of the values of the other

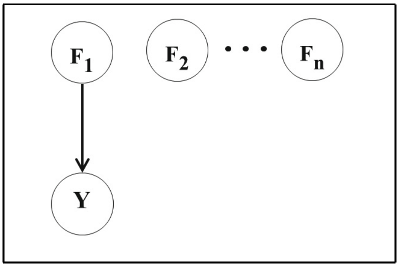


Fig. 2. Illustrating the problem of spurious rules

features: $Y \perp\!\!\!\perp F_i : i \in \{2, ..., n\}$. Assume that pattern $P : F_1 = 1$ is predictive of class $Y = y_1$, so that $Pr(y_1|P) > Pr(y_1)$. Clearly, P is the only important pattern for predicting y_1.

Now consider pattern P' that is a superpattern of P, $P' : F_1 = 1 \wedge F_{q_1} = v_{q_1} \wedge ... \wedge F_{q_k} = v_{q_k}$, where $F_{q_i} \in \{F_2, ..., F_n\}$ and v_{q_i} is *any possible value* of variable F_{q_i}. The network structure implies that $Pr(y_1 \,|\, P') = Pr(y_1 \,|\, P)$, hence $Pr(y_1 \,|\, P')$ is also larger than the prior $Pr(y_1)$.

The problem is that if we evaluate the rules individually (without considering the nested structure of the patterns), we may falsely think that $P' \Rightarrow y_1$ is an important rule. However, this rule is totally redundant given its subrule $P \Rightarrow y_1$. Even by requiring complex rules to have a higher confidence than their simplifications (the confidence improvement) [3, 15, 19, 20, 26], the problem still exists and many spurious rules can easily satisfy this constraint due to noise in sampling. Clearly, having spurious rules in the results is undesirable because they overwhelm the user and prevent him/her from understanding the real causalities in the data.

3 Mining Predictive and Non-Spurious Rules

In this section, we present our approach for scoring/ranking rules. We start by defining a Bayesian score to evaluate the predictiveness of a rule with respect to a more general population. After that, we introduce the *PNSR-score* to address the problem of spurious rules. Lastly, we present an efficient mining algorithm that integrates rule evaluation with frequent pattern mining.

3.1 Classical Rule Quality Measures

A large number of rule quality measures have been proposed in the literature to evaluate the interestingness of individual rules. Examples of such measures include confidence, lift, weighted relative accuracy, J-measure, and others (cf [14]). Most of these measures trade-off two factors: 1) the *distributional unusualness* of the class variable in the rule compared to the general population and 2) the *coverage* of the rule, which reflects its generality [18, 23]. This trade-off is often achieved in an ad-hoc way, for instance by simply multiplying these two factors as in the weighted relative accuracy score [17] or in the J-measure [25]. Furthermore, most interestingness measures rely on point estimates of these quantities, often using the maximum likelihood estimation, and they do not capture the uncertainty of the estimation. In the following, we present a novel Bayesian score to evaluate the quality of a rule.

3.2 The Bayesian Score

Suppose we want to evaluate rule $P \Rightarrow y$ with respect to a group of instances G where $G_P \subseteq G$. Intuitively, we would like the rule to get a high score when there is a strong evidence in the data to support the hypothesis that $Pr(Y=y|G_P) > Pr(Y=y|G)$. Our Bayesian score treats these probabilities as random variables as opposed to using their point estimation as in the classical measures [14].

Let us begin by defining M_e to be the model that conjectures that all instances in group G have the **same probability** for having class $Y=y$, even though we are uncertain what that probability is. Let us denote $Pr(Y=y|G)$ by θ. To represent our uncertainty about θ, we use a beta distribution with parameters α and β. Let N_{*1} be the number of instances in G with class $Y=y$ and let N_{*2} be the number of instances in G with class $Y \neq y$. The marginal likelihood for model M_e is as follows:

$$Pr(G|M_e) = \int_{\theta=0}^{1} \theta^{N_{*1}} \cdot (1-\theta)^{N_{*2}} \cdot beta(\theta;\alpha,\beta)d\theta$$

The above integral yields the following well known closed-form solution [16]:

$$Pr(G|M_e) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha+N_{*1}+\beta+N_{*2})} \cdot \frac{\Gamma(\alpha+N_{*1})}{\Gamma(\alpha)} \cdot \frac{\Gamma(\beta+N_{*2})}{\Gamma(\beta)} \quad (1)$$

where Γ is the gamma function.

Let us define M_h to be the model that conjectures that the probability of $Y=y$ in G_P, denoted by θ_1, is different from the probability of $Y=y$ in the instances of G not covered by P $(G \setminus G_P)$, denoted by θ_2. Furthermore, M_h believes that θ_1 is **higher** than θ_2. To represent our uncertainty about θ_1, we use a beta distribution with parameters α_1 and β_1, and to represent our uncertainty about θ_2, we use a beta distribution with parameters α_2 and β_2. Let N_{11} and N_{12} be the number of instances in G_P with $Y=y$ and with $Y \neq y$, respectively. Let N_{21} and N_{22} be the number of instances outside G_P with $Y=y$ and with $Y \neq y$, respectively (see Figure 3). Note that $N_{*1} = N_{11} + N_{21}$ and $N_{*2} = N_{12} + N_{22}$.

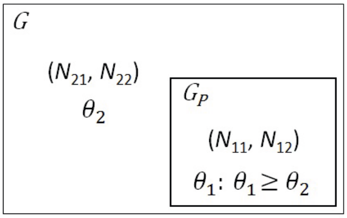


Fig. 3. A diagram illustrating model M_h

The marginal likelihood for model M_h is defined as follows:

$$Pr(G|M_h) = \int_{\theta_1=0}^{1} \int_{\theta_2=0}^{\theta_1} {\theta_1}^{N_{11}} \cdot (1-\theta_1)^{N_{12}} \cdot {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot \frac{beta(\theta_1;\alpha_1,\beta_1)\cdot beta(\theta_2;\alpha_2,\beta_2)}{k} d\theta_2 d\theta_1 \tag{2}$$

where k is a normalization constant for the parameter prior[1]. Note that this formula does not assume that the parameters are independent, but rather constrains θ_1 to be higher than θ_2.

Below we show the closed form solution we obtained by solving Equation 2. The derivation of the solution is omitted in this manuscript due to space limitation[2].

$$Pr(G|M_h) = \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\Gamma(\beta_2)} \cdot \sum_{j=a}^{a+b-1} \left(\frac{\Gamma(a)\Gamma(b)}{\Gamma(j+1)\Gamma(a+b-j)} \cdot \frac{\Gamma(c+j)\Gamma(a+b+d-j-1)}{\Gamma(a+b+c+d-1)} \right) \tag{3}$$

where $a = N_{21} + \alpha_2$, $b = N_{22} + \beta_2$, $c = N_{11} + \alpha_1$, $d = N_{12} + \beta_1$. We solve for k by applying Equation 3 (without the k term) with $a = \alpha_2$, $b = \beta_2$, $c = \alpha_1$ and $d = \beta_1$.

Equation 3 can be expressed in logarithmic form (to avoid computing very large numbers). Its computational complexity is $O(b) = O(N_{22}+\beta_2)$ (the number of terms in the summation). It turns out that we can redefine the solution of Equation 2 so that its computational complexity is $O(min(N_{11}+\alpha_1, N_{12}+\beta_1, N_{21}+\alpha_2, N_{22}+\beta_2))$. The modifications that achieve this complexity result are omitted due to space limitation[3].

Lastly, let M_l be the model that conjectures that θ_1 is **lower** than θ_2. The marginal likelihood for M_l is similar to Equation 2, but integrates θ_2 from 0 to 1 and constrains θ_1 to be integrated from 0 to θ_2 (forcing θ_1 to be smaller than θ_2). The solution for $P(G|M_l)$ can reuse the terms computed in Equation 3 and can be computed with complexity O(1).

Now that we computed the marginal likelihood for models M_e, M_h and M_l, we compute the posterior probability of M_h (the model of interest) using Bayes theorem:

$$Pr(M_h|G) = \frac{Pr(G|M_h)Pr(M_h)}{Pr(G|M_e)Pr(M_e)+Pr(G|M_h)Pr(M_h)+Pr(G|M_l)Pr(M_l)} \tag{4}$$

To be "non-informative", we might simply assume that all three models are equally likely a-priori: $Pr(M_e) = Pr(M_h) = Pr(M_l) = \frac{1}{3}$.

Equation 4 quantifies in a Bayesian way how likely (a posteriori) is the model which presumes $Pr(Y = y|G_P)$ is higher than $Pr(Y = y|G)$. Since this is the

[1] $k = \frac{1}{2}$ if we use uniform priors on both parameters by setting $\alpha_1 = \beta_1 = \alpha_2 = \beta_2 = 1$.

[2] The derivation can be found on the author's website: www.cs.pitt.edu/~iyad.

[3] These modifications can found on the author's website: www.cs.pitt.edu/~iyad.

quantity we are interested in, we use $Pr(M_h|G)$ to score rule $P \Rightarrow y$ with respect to group G. We denote this Bayesian score by $\boldsymbol{BS(P \Rightarrow y, G)}$.

Example 2. Let us use the Bayesian score to evaluate rule R_2: Family history=Yes ⇒ CHD in Example 1. We evaluate R_2 with respect to the entire dataset G_ϕ by computing $BS(R_2, G_\phi)$. Using the notations introduced earlier, $N_{*1} = 50$ and $N_{*2} = 100$ (the number of cases and controls in the dataset). Also, $N_{11} = 30$, $N_{12} = 20$, $N_{21} = N_{*1} - N_{11} = 20$ and $N_{22} = N_{*2} - N_{12} = 80$. Let us use uniform beta priors for all parameters: $\alpha = \beta = \alpha_1 = \beta_1 = \alpha_2 = \beta_2 = 1$. The likelihood of M_e is 3.2×10^{-43}, the likelihood of M_h is 1.5×10^{-38} and the likelihood of M_l is 1×10^{-44}. Hence, $BS(R_2, G_\phi) = Pr(M_h|G_\phi) = 0.99998$. This implies that there is a strong evidence in the data to conclude that pattern Family history=yes makes CHD more likely.

3.3 The Predictive and Non-Spurious Rules Score

The Bayesian score proposed in the previous section provides a way to evaluate the predictiveness of a rule by contrasting it to a more general population than the population covered by the rule. One approach to supervised descriptive rule discovery is to score each rule R_i with respect to the entire data $BS(R_i, G_\phi)$ and report the top rules to the user. However, this approach does not overcome the spurious rules problem: if a rule $P \Rightarrow y$ achieves a very high score, many spurious rules $P' \Rightarrow y$: $P' \supset P$ are expected to have a high score as well (provided that P' have enough support in the data). As a result, the rules presented to the user would contain a lot of redundancies and fail to provide a good coverage of the data.

To overcome this problem, we propose the *Predictive and Non-Spurious Rules score*, denoted as *PNSR-score*, which we define as follows:

$$\textit{PNSR-score}(P \Rightarrow y) = \min_{S: S \subset P} \{BS(P \Rightarrow y, G_S)\}$$

If a rule R achieves a high *PNSR-score*, then there is a strong evidence in the data not only to conclude that R improves the prediction of its consequent with respect to the entire data, but also with respect to the data matching any of its subrules. That is, the rule's effect on the class distribution cannot be explained by any more general rule that covers a larger population. This implies that every item in the condition of the rule is an important contributor to its predictiveness (the rule is concise).

Example 3. Let us go back to Example 1 and compute the *PNSR-score* for rule R_3. If we evaluate R_3 with respect the entire dataset, $BS(R_3, G_\phi) = 0.9997$. If we evaluate R_3 with respect to subrule R_1, $BS(R_3, G_{R_1}) = 0.999992$. Finally, if we evaluate R_3 with respect to subrule R_2, $BS(R_3, G_{R_2}) = 0.47$. We can see that R_3 is considered very predictive when compared to the entire dataset or to subrule R_1, but is not predictive when compared to subrule R_2. Therefore, we do not consider R_3 an important rule because it is equivocal whether it predicts CHD as being more likely than does R_2.

Example 4. Let us consider again the simple Bayesian network in Figure 2. Assume we have 10 binary features (F_1 to F_{10}) and the CPTs are defined as follows: $Pr(F_i = 1) = 0.4 : i \in \{1, ..., 10\}$, $Pr(Y = y_1|F_1 = 1) = 0.9$ and $Pr(Y = y_1|F_1 = 0) = 0.5$. Let the data D be 500 instances that are randomly generated from this network and let us use D to mine rules that are predictive of class y_1[4]. As we discussed earlier, the only important rule for predicting y_1 is $F_1 = 1 \Rightarrow y_1$ and all other rules are just spurious.

We use frequent pattern mining to explore all patterns that occur in more than 10% of the data. Doing so, we obtain 1,257 frequent patterns (potential rules). If we apply the χ^2 test with significance level $\alpha = 0.05$, we get 284 rules that positively predicts y_1 and are statistically significant. Even if we apply the False Discovery Rate (FDR) technique [4] to correct for multiple hypothesis testing, we get 245 positive significant rules! If we use our Bayesian score to evaluate each rule (individually) with respect to the entire dataset and report rules with $BS(R_i, G_\phi) \geq 0.95$, we get 222 rules[5]. Note that this approach still suffers from the spurious rules problem. Let us now apply the confidence improvement constraint to filter out "non-productive" rules [3, 15, 19, 20, 26]. By doing so, we get 451 rules! This clearly demonstrates that the confidence improvement constraint is ineffective for removing spurious rules. Lastly, let us use our proposed *PNSR-score* and report rules with *PNSR-score*$(R_i) \geq 0.95$. Doing so, we obtain only a single rule $F_1 = 1 \Rightarrow y_1$ (the only important rule) and effectively filter out all other spurious rules.

3.4 The Mining Algorithm

In this section, we present the algorithm for mining predictive and non-spurious rules. The algorithm utilizes frequent pattern mining to explore the space of potential rules and applies the *PNSR-score* to evaluate the rules.

To search for rules, we partition the data according to the class labels $y \in dom(Y)$ and mine frequent patterns for each class separately (using a local minimum support σ_y that is related to the number of instances from class y). The reason for doing this as opposed to mining frequent patterns from the entire data is that when the data is unbalanced, exploring only patterns that are globally frequent may result in missing many important rules for the rare classes.

The mining algorithm takes as input 1) the data instances from class y: $D_y = \{(x_i, y_i) : y_i = y\}$, 2) the data instances that do not belong to class y: $D_{\neg y} = \{(x_i, y_i) : y_i \neq y\}$, 3) the local minimum support threshold σ_y and 4) a user specified significance parameter g. The algorithm explores the space of frequent patterns and outputs the rules with *PNSR-score* higher than g.

A straightforward way to obtain the result is to apply the commonly used two-phase approach as in [6, 10, 12, 17, 20, 26, 27], which generates all frequent patterns in the first phase and evaluates them in the second phase (a post-processing phase). That is, we need to perform the following two steps:

[4] The prior of y_1 in this network is $Pr(Y = y_1) = 0.66$.

[5] The 0.95 threshold is chosen so that it is comparable to the commonly used frequentist 0.05 significance level.

1. Phase I: Mine all frequent patterns: $FP = \{P_1, ..., P_m : sup(P_i) \geq \sigma_y\}$
2. Phase II: For each pattern $P_i \in FP$, output rule $P_i \Rightarrow y$ if $\mathit{PNSR\text{-}score}(P_i \Rightarrow y) \geq g$.

In contrast to this two-phase approach, our algorithm integrates rule evaluation with frequent pattern mining, which allows us to apply additional pruning techniques that are not applicable in the two-phase approach.

The mining algorithm explores the lattice of frequent patterns level by level from the bottom-up starting from the empty pattern. That is, the algorithm first explores frequent *1-patterns*, then frequent *2-patterns*, and so on. When the algorithm visits a frequent pattern P (a node in the lattice), it computes the *PNSR-score* of rule $P \Rightarrow y$ and adds it to result if $\mathit{PNSR\text{-}score}(P \Rightarrow y) \geq g$.

Lossless Pruning. We now illustrate how to utilize the *PNSR-score* to prune portions of the search space that are guaranteed not to contain any result.

We say that pattern P is **pruned** if we do not explore any of its superpatterns ($P' \supset P$). This means that we exclude the entire sublattice with bottom P from the lattice of patterns we have to explore.

Frequent pattern mining relies only on the *support* of the patterns to prune infrequent patterns according to the following anti-monotone property: if a pattern is not frequent, all of its superpatterns are guaranteed not to be frequent.

By integrating rule evaluation with frequent pattern mining, we can apply an additional pruning technique. The idea is to prune pattern P if we guarantee that none of its superpatterns will be in the result:

$$\mathit{Prune}\ P \text{ if } \forall P' \supset P : \mathit{PNSR\text{-}score}(P' \Rightarrow y) < g$$

However, since patterns are explored in a level-wise fashion, we do not know the class distribution in the superpatterns of P. But we know that for any $P' \supset P$: $G_{P'} \subseteq G_P$, and hence $sup(P', D_y) \leq sup(P, D_y) \wedge sup(P', D_{\neg y}) \leq sup(P, D_{\neg y})$.

We now define the *optimal superpattern* of P with respect to class y, denoted as P^*, to be a hypothetical pattern that covers all instances from y and none of the instances from the other classes:

$$sup(P^*, D_y) = sup(P, D_y) \wedge sup(P^*, D_{\neg y}) = 0$$

P^* is the best possible superpattern for predicting y that P can generate. Therefore, $\mathit{PNSR\text{-}score}(P^* \Rightarrow y)$ is an upper bound on the *PNSR-score* for the superpattern of P. Now, we safely prune P when $\mathit{PNSR\text{-}score}(P^* \Rightarrow y) < g$.

4 Experimental Evaluation

The experiments compare the performance of different rule quality measures for the problem of supervised descriptive rule discovery. In particular, we compare the following measures:

1. *GR*: Rules are ranked using the Growth Rate measure, which was used in [11] in the context of emerging pattern mining.

$$GR(P \Rightarrow y) = \frac{sup(P, D_y)/|D_y|}{sup(P, D_{\neg y})/|D_{\neg y}|}$$

   where D_y and $D_{\neg y}$ represent the instances from class y and not from class y, respectively.
2. *J-measure*: Rules are ranked using the J-measure [25], a popular information theoretic measure that scores the rules by their information content.

$$J\text{-}measure(P \Rightarrow y) = \frac{sup(P, D)}{|D|} \times \sum_{z \in \{y, \neg y\}} conf(P \Rightarrow z) \cdot log_2 \left(\frac{conf(P \Rightarrow z)}{conf(\Phi \Rightarrow z)} \right)$$

3. *WRAcc*: Rules are ranked using the Weighted Relative Accuracy, which was used in [17] in the context of subgroup discovery[6].

$$WRAcc(P \Rightarrow y) = \frac{sup(P, D)}{|D|} \times (conf(P \Rightarrow y) - conf(\Phi \Rightarrow y))$$

   Note that this measure is compatible (provides the same rule ranking) with the support difference heuristic used in [2] for contrast set mining (see [23]).
4. *BS*: Rules are ranked using our proposed Bayesian score. However, this method scores each rule individually with respect to the entire data and do not filter out spurious rules.
5. *Conf-imp*: Only rules that satisfy the confidence improvement constraint are retained [3, 15, 19, 20, 26] and they are ranked according to their confidence.
6. *PNSR*: Only rules R_i that have a $PNSR\text{-}score(R_i) \geq 0.95$ are retained[7] and they are ranked according to the Bayesian score.

Note that the *GR* measure does not consider the coverage of the rule when assessing its interestingness. For example, *GR* favors a rule that covers 8% of the instances of in one class and 1% of the instances in the other classes over a rule that covers 70% of the instances of in one class and 10% of the instances in the other classes (as $\frac{8}{1} > \frac{70}{10}$). As a result, *GR* often chooses rules that are very specific (with low coverage) and do not generalize well. To overcome this, the *J-measure* and *WRAcc* explicitly incorporate the rule coverage $\frac{Sup(P,D)}{|D|}$ in their evaluation functions to favor high coverage rules over low coverage rules. This is done by multiplying the rule coverage with a factor that quantifies the distributional surprise (unusualness) of the class variable in the rule (the cross entropy for *J-measure* and the relative accuracy for *WRAcc*). However, it is not clear whether simply multiplying these two factors leads to the optimal trade-off. On the other hand, *BS* achieves this trade-off automatically by modeling the

[6] The algorithm by [17] uses weighted sequential covering and modifies the WRAcc measure to handel example weights.

[7] The 0.95 threshold is chosen so that it is comparable to the commonly used frequentist 0.05 significance level.

uncertainty of the estimation (the more data we have, the more certain we are about the estimated probabilities).

Note that the first four methods (*GR*, *J-measure*, *WRAcc* and *BS*) evaluate each rule individually with respect to the entire data and do not consider the nested structure of rules. On the other hand, *conf-imp* and *PNSR* evaluate each rule with respect to all of its subrules. *Conf-imp* simply requires each rule have a higher confidence than its subrules, while *PNSR* requires each rule to show a substantial evidence that it improves the prediction over its subrules, which is evaluated using our proposed *PNSR-score*.

For all methods, we use frequent pattern mining to explore the space of potential rules and we set the local minimum support (σ_y) to 10% the number of instance in the class. For *BS* and *PNSR*, we use uniform beta priors (uninformative priors) for all parameters.

4.1 Datasets

We evaluate the performance of the different rule quality measures on 15 public datasets from the UCI Machine Learning repository. We discretize the numeric attributes into intervals using Fayyad and Irani discretization [13]. Table 1 shows the main characteristics of the datasets.

Table 1. UCI Datasets characteristics

dataset	# features	# instances	# classes
Lymphography	18	142	2
Parkinson	22	195	2
Heart	13	270	2
Hepatitis	19	155	2
Diabetes	8	768	2
Breast cancer	9	286	2
Nursery	8	12,630	3
Red wine	11	1,599	3
Mammographic	5	961	2
Tic tac toe	9	958	2
Ionosphere	34	351	2
Kr vs kp	36	3,196	2
Pen digits	16	10,992	10
Zoo	16	74	3
WDBC	30	569	2

4.2 Quality of Top-K Rules

For a set of rules to be practically useful, the rules should be accurate to predict the class label of unseen data instances (high precision) and the rule set should provide a good coverage of the data (high recall).

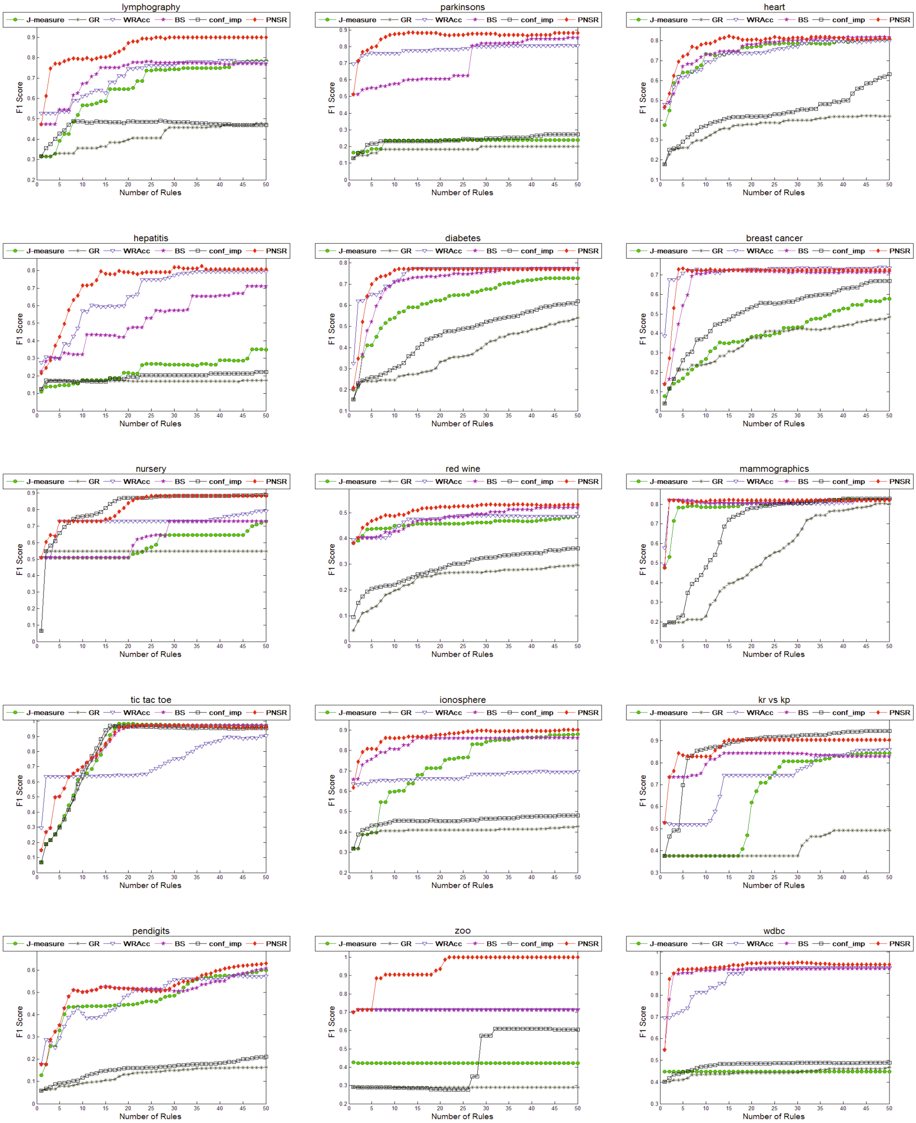


Fig. 4. Comparison of the performance of several rule evaluation measures. The X-axis is the number of the top rules and the Y-axis is the F1 score of the rule set.

In this section, we compare the different rule evaluation measures according to the quality of the top rules. In particular, for each of the compared evaluation measures, we mine the top k rules from the training data and use them to classify the testing data. The classification is done according to the highest confidence rule [21]:

$$\mathit{Prediction}(x) = \arg\max_{y_i}\{\mathit{conf}(P \Rightarrow y_i)\colon P \in x\}$$

The classification performance is evaluated using the F1 score [24], which is the harmonic mean of the precision and recall. All results are reported using a 10-fold cross-validation scheme, where we use the same train/test splits for all compared methods.

Figure 4 shows the classification performance for the different number of top rules. We can see that *GR* is the worst performing method for most datasets. The reason is that rules with the highest *GR* scores are usually very specific (low coverage) and may easily overfit the training data. The other measures (*J-measure*, *WRAcc* and *BS*) perform better than *GR* because they favor high-coverage rules over low-coverage rules, which results in rules that generalize better on the testing data. However, because these measures do not consider the relations among the rules, the top rules contain many spurious rules (rules describing the same underlying pattern and are small variations of each other). As a result, they fail to provide a good coverage of the data (see for example the *lymphography* and the *zoo* datasets). Finally, we can see that for most datasets, *PNSR* achieves the best performances with the smallest number of rules.

4.3 Mining Efficiency

In this section, we study the efficiency of our mining algorithm. In particular, we compare the running time of the following methods:

1. *FPM*: Frequent patterns mining, where we partition the data according to the class label and mine frequent patterns for each class (see Section 3.4). We apply the algorithm by [29], which mines frequent patterns using the vertical data format.
2. *PNSR-Naive*: The naive two-phase implementation for mining predictive and non-spurious rules, which applies *FPM* to mine all frequent patterns and then computes the *PNSR-score* of the patterns.
3. *PNSR*: Our mining algorithm, which integrates rule evaluation with frequent pattern mining and applies the lossless pruning technique described in Section 3.4 to prune the search space.

The running time is measured on a Dell Precision T1600 machine with an Intel Xeon 3GHz CPU and 16GB of RAM. As before, we set the local minimum support (σ_y) to 10% the number of instance in the class. Table 2 shows the execution time (in seconds) of the compared methods on the UCI datasets.

The results show that on seven of the fifteen datasets (*lymphography*, *Parkinson*, *Heart*, *Hepatitis*, *Ionosphere*, *Zoo* and *WDBC*), *PNSR* is more efficient than *FPM*, which is the cost of the first phase of any two-phase method

Table 2. The execution time (in seconds) of frequent pattern mining (*FPM*), two-phase PNSR mining (*PNSR-Naive*) and our mining algorithm (*PNSR*)

dataset	*FPM*	*PNSR-Naive*	*PNSR*
Lymphography	328	410	153
Parkinson	9,865	11,229	800
Heart	45	69	37
Hepatitis	1,113	1,284	391
Diabetes	3	5	5
Breast cancer	3	5	4
Nursery	2	9	9
Red wine	28	52	50
Mammographic	1	1	1
Tic tac toe	3	4	4
Ionosphere	16,899	19,765	1,077
Kr vs kp	1,784	2,566	2,383
Pen digits	71	144	138
Zoo	185	244	23
WDBC	2,348	4,320	282

[6, 10, 12, 17, 20, 26, 27]. For some of these datasets, *PNSR* drastically improves the efficiency. For example, on the *Parkinson*, *Ionosphere* datasets, *PNSR* is more than an order of magnitude faster than *FPM*. This shows that utilizing the predictiveness of patterns to prune the search space can greatly help improving the mining efficiency.

5 Conclusion

In this paper, we study the problem of supervised descriptive rule discovery and propose a new rule evaluation score, the Predictive and Non-Spurious Rules (PNSR) score. This score relies on Bayesian inference to measure the quality of the rules. It also considers the structure of the patterns to ensure that each rule in the result offers a significant predictive advantage over all of its generalizations. We present an algorithm for mining rules with high PNSR scores, which efficiently integrates rule evaluation with frequent pattern mining. The experimental evaluation shows that our method is able to explain and cover the data with fewer rules than existing methods, which is beneficial for knowledge discovery.

Acknowledgments. This work was supported by grants 1R01GM088224-01 and 1R01LM010019-01A1 from the NIH. Its content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the International Conference on Very Large Data Bases, VLDB (1994)

2. Bay, S.D., Pazzani, M.J.: Detecting group differences: Mining contrast sets. Data Mining and Knowledge Discovery 5, 213–246 (2001)
3. Bayardo, R.J.: Constraint-based rule mining in large, dense databases. In: Proceedings of the International Conference on Data Engineering, ICDE (1999)
4. Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: A practical and powerful approach to multiple testing. Journal of the Royal Statistical Society 57(1), 289–300 (1995)
5. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: generalizing association rules to correlations. In: Proceedings of the International Conference on Management of Data, SIGMOD (1997)
6. Cheng, H., Yan, X., Han, J., Wei Hsu, C.: Discriminative frequent pattern analysis for effective classification. In: Proceedings of the International Conference on Data Engineering, ICDE (2007)
7. Clark, P., Niblett, T.: The cn2 induction algorithm. Machine Learning (1989)
8. Cohen, W.: Fast effective rule induction. In: Proceedings of International Conference on Machine Learning, ICML (1995)
9. Cohen, W., Singer, Y.: A simple, fast, and effective rule learner. In: Proceedings of the National Conference on Artificial Intelligence, AAAI (1999)
10. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. IEEE Transactions on Knowledge and Data Engineering 17, 1036–1050 (2005)
11. Dong, G., Li, J.: Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, SIGKDD (1999)
12. Exarchos, T.P., Tsipouras, M.G., Papaloukas, C., Fotiadis, D.I.: A two-stage methodology for sequence classification based on sequential pattern mining and optimization. Data and Knowledge Engineering 66, 467–487 (2008)
13. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI (1993)
14. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Computing Surveys 38 (2006)
15. Grosskreutz, H., Boley, M., Krause-Traudes, M.: Subgroup discovery for election analysis: a case study in descriptive data mining. In: Proceedings of the International Conference on Discovery Science (2010)
16. Heckerman, D., Geiger, D., Chickering, D.M.: Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning (1995)
17. Kavsek, B., Lavrač, N.: APRIORI-SD: Adapting association rule learning to subgroup discovery. Applied Artificial Intelligence 20(7), 543–583 (2006)
18. Lavrač, N., Gamberger, D.: Relevancy in Constraint-Based Subgroup Discovery. In: Boulicaut, J.-F., De Raedt, L., Mannila, H. (eds.) Constraint-Based Mining. LNCS (LNAI), vol. 3848, pp. 243–266. Springer, Heidelberg (2006)
19. Li, J., Shen, H., Topor, R.: Mining Optimal Class Association Rule Set. In: Cheung, D., Williams, G.J., Li, Q. (eds.) PAKDD 2001. LNCS (LNAI), vol. 2035, p. 364. Springer, Heidelberg (2001)
20. Li, W., Han, J., Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In: Proceedings of the International Conference on Data Mining, ICDM (2001)
21. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Knowledge Discovery and Data Mining, pp. 80–86 (1998)

22. Nijssen, S., Guns, T., De Raedt, L.: Correlated itemset mining in roc space: a constraint programming approach. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, SIGKDD (2009)
23. Novak, P.K., Lavrač, N., Webb, G.I.: Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. Journal of Machine Learning Research (JMLR) 10, 377–403 (2009)
24. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys (2002)
25. Smyth, P., Goodman, R.M.: An information theoretic approach to rule induction from databases. IEEE Transactions on Knowledge and Data Engineering (1992)
26. Webb, G.I.: Discovering significant patterns. Machine Learning 68(1), 1–33 (2007)
27. Xin, D., Cheng, H., Yan, X., Han, J.: Extracting redundancy-aware top-k patterns. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, SIGKDD (2006)
28. Yang, Y., Webb, G.I., Wu, X.: Discretization methods. In: The Data Mining and Knowledge Discovery Handbook, pp. 113–130. Springer (2005)
29. Zaki, M.J.: Scalable algorithms for association mining. IEEE Transaction on Knowledge and Data Engineering (TKDE) 12, 372–390 (2000)

Generic Pattern Trees for Exhaustive Exceptional Model Mining

Florian Lemmerich[1], Martin Becker[1], and Martin Atzmueller[2]

[1] Artificial Intelligence and Applied Computer Science Group,
University of Würzburg, D-97074 Würzburg, Germany
{lemmerich,becker}@informatik.uni-wuerzburg.de
[2] Knowledge & Data Engineering Group,
University of Kassel, 34121 Kassel, Germany
atzmueller@cs.uni-kassel.de

Abstract. Exceptional model mining has been proposed as a variant of subgroup discovery especially focusing on complex target concepts. Currently, efficient mining algorithms are limited to heuristic (non exhaustive) methods. In this paper, we propose a novel approach for fast exhaustive exceptional model mining: We introduce the concept of valuation bases as an intermediate condensed data representation, and present the general GP-growth algorithm based on FP-growth. Furthermore, we discuss the scope of the proposed approach by drawing an analogy to data stream mining and provide examples for several different model classes. Runtime experiments show improvements of more than an order of magnitude in comparison to a naive exhaustive depth-first search.

Keywords: exceptional model mining, subgroup discovery.

1 Introduction

Subgroup discovery [9,17] has been established as a general and broadly applicable technique for descriptive data mining: It aims at identifying descriptions of subsets of a dataset that show an interesting behavior with respect to a certain *target property of interest*. In this context, the concept of *exceptional model mining* has been introduced [6,13], which especially focuses on complex target properties: It tries to identify interesting patterns with respect to a local model derived from *a set* of attributes. The interestingness can be defined, e.g., by a significant deviation from a model that is derived from the total population or the respective complement set of instances within the population. While there exist heuristic algorithms [11] for exceptional model mining, the efficient exhaustive computation of exceptional models is still an open issue.

In this paper, we present the novel *GP-growth* algorithm that can be used for mining patterns with exceptional target models *exhaustively*. We extend the well-known FP-tree [7] data structure by replacing the frequency information stored in each node of the tree by the more general concept of valuation bases. Valuation bases are dependent on a specific *model class* and allow for an efficient computation of the target model parameters.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 277–292, 2012.

The contribution of this paper is threefold: First, we present the concept of valuation bases allowing us to derive a new algorithm capable of performing efficient exhaustive search for many different classes of exceptional models. Second, we characterize the scope of the presented approach and discuss its instantiations for model classes presented in literature. Third, we perform an evaluation of the presented approach using publicly available UCI data [14]. Furthermore, we present a scalability study on a real world dataset.

The remainder of the paper is structured as follows: Section 2 discusses related work. Section 3 introduces the formal background of exceptional model mining and briefly reviews the FP-growth algorithm. Next, Section 4 presents the novel GP-growth algorithm. Instantiations for different model classes are discussed in Section 5. After that, we demonstrate the effectiveness and validity of our approach using publicly available data in Section 6. Finally, the paper concludes with a summary and outlook on future work in Section 7.

2 Related Work

For a recent overview on algorithms for subgroup discovery [9,17], including heuristic and exhaustive approaches, we refer to [8]. Kralj et al. also discuss subgroup discovery in relation to other common approaches for supervised descriptive rule discovery [15]. Exceptional model mining has been proposed in [6,13]. [11] presents an heuristic algorithm for identifying descriptions of exceptional subgroups. Umek et al. describe subgroup discovery in a similar setting [16]. Again, their proposed algorithm does not perform an exhaustive search.

To the best of the authors' knowledge, no non-trivial exhaustive algorithm for the task of exceptional model mining has been published so far. To this end, we propose the GP-growth algorithm based on FP-growth and FP-trees. Originally developed for association rule mining [7], FP-trees are a widely-used data representation. It has been adapted to subgroup discovery with nominal [3] and numeric target concepts [1]. The transfer to exceptional model mining is not trivial but – as shown in this paper – possible for many model classes.

3 Background

In the following, we first provide a brief summary of exceptional model mining [6,13]. After that, we review the FP-growth algorithm.

3.1 Exceptional Model Mining

Exceptional model mining aims at identifying models (patterns) that are *exceptional* concerning a set of target attributes. For some basic definitions, a database $D = (I, A)$ is given by a set of data instances (cases) $i \in I$ and a set of attributes A. The attributes consist of two (usually non-overlapping) sets of describing attributes A_D and model attributes A_M. We denote the value of attribute X in instance i as i_X. *Selectors* or *basic patterns* are boolean functions $I \to \{\mathit{false}, \mathit{true}\}$ defined by selection expressions on the set of describing attributes A_D. Typical selection expression are given by attribute-value pairs in the case of nominal attributes, or by intervals in the case of numeric attributes. The selector $\mathit{age} = [12; 18]$ is true, for example, iff the attribute *age* has a value between 12 and 18 for the respective instance. A *subgroup description* or (complex) *pattern* combines selectors into a boolean formula. For a typical conjunctive description language, a pattern $p = \{\mathrm{sel}_1, \ldots, \mathrm{sel}_k\}$ is defined by a set of selectors sel_j, which are interpreted as a conjunction, i.e. $p = \mathrm{sel}_1 \wedge \ldots \wedge \mathrm{sel}_k$. A *subgroup* corresponding to a pattern p contains all instances for which p evaluates to *true*.

A model consists of a specific *model class*, which is fixed for a specific mining task, and *model parameters* which depend on the values of the model attributes in the instances of the respective subgroup. The goal of exceptional model mining is then to identify descriptions of subgroups, for which the model parameters differ significantly from the parameters of the model built from the entire dataset. Formally this is accomplished by using an *exceptionality measure* q that maps a subgroup (pattern) to a real number corresponding to its quality (interestingness) based on its model parameters. As a simple example, consider the task of identifying subgroups in which the correlation between two numeric attributes is especially strong. This *correlation model class* has exactly one parameter, i.e., the correlation coefficient. A short overview on different model classes presented in literature is included in Section 5.

To accomplish the exceptional model mining task for a set of l selectors there are $O(2^l)$ subgroup descriptions, for which the model parameters need to be determined. For practical purposes it is often possible to limit the search space to patterns with a maximum number d of contained selectors ($|p| \leq d$) (since longer patterns are difficult to interpret by humans). However, identifying the best pattern is still challenging since the size of the search space is still $O(l^d)$. In this paper, we provide an efficient exhaustive algorithm for this task.

3.2 FP-Growth

The FP-growth [7] algorithm has been introduced as an efficient approach for frequent pattern mining. It avoids scanning the whole dataset in order to evaluate each pattern by recursively building a special data structure, the so-called FP-tree. This extended prefix tree structure contains the relevant data in a compressed way. Each tree node contains a reference to a selector and a frequency count. Additionally, links between nodes referring to the same selector are maintained. The FP-tree is built by sorting the selectors of each data instance according to their descending frequency in the dataset. Then, each data instance is inserted into the FP-tree. The order of the selectors increases the chance of shared prefixes between data instances decreasing the size of the FP-tree. Most importantly, the resulting FP-tree contains the complete condensed frequency information for each data instance.

FP-growth starts with creating an FP-tree for the initial dataset. Patterns containing exactly one selector are evaluated by the frequencies collected during the first pass over the dataset. Then, the algorithm recursively extends those patterns by adding further selectors in a depth-first manner, building conditional trees conditioned on the current pattern prefix. Each node corresponds to a conditional data instance built from the selectors referred to by its parent nodes. In this way, FP-growth enables a compact and efficient mining of the condensed tree structure. Due to the limited space, we refer to [7] for more details.

4 The GP-Growth Algorithm

In this section, we present our novel approach called *generic pattern growth (GP-growth)*. GP-growth is based on the FP-growth algorithm substituting frequencies by an intermediate, condensed data representation called a valuation basis. We first introduce the concept of valuation bases and define properties needed for their application to the GP-growth algorithm. After that, we provide a theorem concerning the existence and construction of efficient valuation bases for specific model classes.

4.1 The Concept of Valuation Bases

In the traditional FP-growth approach frequencies are stored in the nodes of the tree. These nodes can then be aggregated to obtain frequencies for patterns in the search space. The frequency is then used to rate the patterns (itemsets) determining those with high instance counts. In the proposed approach, we replace the frequency count stored in each node of the tree structure with a more generic concept that we call a *valuation basis*.

We define a valuation basis as a (condensed) representation of a set of data instances that is sufficient to extract the model parameters for a given model class. Consequently, the kind of information stored in a valuation basis is dependent on the model class. Since the interestingness of a subgroup in our definition is based on the model parameters, it can be derived from such a valuation basis.

A visualization of the overall approach is given in Figure 1: For each subgroup (set of data instances) a valuation basis can be derived using a function ϕ (*valuation projector*). The model parameters for the chosen model class can be extracted from these valuation bases using another function χ (*model extractor*). Model parameters are then used to determine the interestingness of the respective pattern using an exceptionality measure q.

As an example, consider a very basic model for a single model attribute X, for which the only model parameter is the mean value of all instances covered by the subgroup. Then, an appropriate valuation base can consist of the instance count and the sum of all values of X of all instances of the subgroup. The instance count and the sum of values can be accumulated in an FP-tree like structure. Given the accumulated valuation basis for each pattern, the stored instance count and the value sum are used to compute the mean. The actual interestingness of the pattern can then be determined using this mean value, e.g. as the deviation from the mean value in the total population.

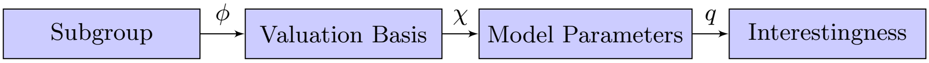


Fig. 1. The pipeline visualizing our approach: For each subgroup (set of data instances) s a valuation basis can be derived using a function ϕ (valuation projector). The model parameters for the chosen model class can be extracted from these valuation bases using another function χ (model extractor). Model parameters are then used to determine the interestingness of the respective pattern using an exceptionality measure q.

Please note, that we can construct a trivial type of valuation bases that defines a valuation basis as the exact same set of data instances it represents (restricted to the model attribute values). Obviously, this most general type of valuation bases trivially contains all relevant information associated with the original set of data instances. Therefore, model parameters for any model class on the original set of data instances can be derived from this type of valuation bases. However, while this trivial kind of valuation basis allows for a general applicable approach, the main advantages in terms of memory and runtime performance are usually lost. Therefore, we aim to construct valuation bases for a given model class which are as small as possible. We call a valuation basis a *condensed valuation basis*, if its memory requirement is sublinear with respect to the number of instances it represents. The examples of valuation bases that we present in this paper will all use constant memory with respect to the instance count.

We model the possibility of aggregating valuation bases corresponding to sets of data instances by introducing the notion of *valuation domains*:

Definition 1 (Valuation Domain, Valuation Basis). *A valuation domain is an abelian semi-group $\mathbb{V} = (V, \oplus)$, where V is an arbitrary set and $\oplus$ is a binary operator on V, i.e. $\oplus : V \times V \to V$ and*

- *V is closed under $\oplus$, i.e. $a, b \in V \Rightarrow a \oplus b \in V$*
- *$\oplus$ is associative, i.e. $a, b, c \in V \Rightarrow a \oplus (b \oplus c) = (a \oplus b) \oplus c$*
- *$\oplus$ is commutative, i.e. $a, b \in V \Rightarrow a \oplus b = b \oplus a$*

An element $v \in V$ is called a valuation basis.

In order to derive valuation bases from data instances we define the notion of a *valuation projector* ϕ as in Definition 2.

Definition 2 (Valuation Projector). *Let I be a set of all data instances and let $\mathbb{V} = (V, \oplus)$ be a valuation domain. Then a valuation projector is defined as*

$$\phi : I \to V$$

Given the definition of valuation domains above, the *valuation projector* ϕ can be naturally extended onto sets of data instances $S \in 2^I$:

$$\begin{aligned} \bar{\phi} : 2^I &\to V \\ S &\mapsto \bigoplus_{s \in S} \phi(s) \end{aligned}$$

As a result, for any disjunct pair of sets of data instances $S' \cap S'' = \emptyset$ it holds:

$$\bar{\phi}(S' \cup S'') = \bar{\phi}(S') \oplus \bar{\phi}(S'')$$

Sometimes patterns are evaluated by comparing their corresponding model attributes derived from the set of data instances they cover with the model attributes of their complementary set of data instances. In order to handle this case efficiently we need to assume a valuation domain with a subtraction operator $\ominus$: Let I be the set of all instances and let $I' \cup I''$ be an arbitrary partition of I. If the valuation basis of I' is subtracted from the valuation basis of I, then the result must be the valuation basis of I'': $\bar{\phi}(I) \ominus \bar{\phi}(I') = \bar{\phi}(I'')$. This can be utilized by computing the valuation basis corresponding to all instances in the dataset I in an initial pass over the dataset.

4.2 Algorithmic Adaptations

Essentially, the FP-growth algorithm (cf. Section 3.2 for a brief description) is generalized by substituting frequencies with the more general concept of valuation bases. We call the resulting algorithm *GP-growth*. The generalized tree structure storing valuation bases instead of frequencies is called a *GP-tree*.

Whereas the FP-growth algorithm adds up frequencies in its FP-trees, the GP-growth algorithm aggregates valuation bases in its GP-trees. Hence, GP-growth produces aggregated valuation bases instead of frequencies for each pattern. Note that a valuation basis can also contain a frequency as described in the mean value example in Section 4.1.

In order to aggregate valuation bases for each pattern, the algorithm requires

- a valuation domain $\mathbb{V} = (V, \oplus)$ to draw valuation bases from,
- a valuation projector ϕ to project single data instances onto valuation bases, and
- a subtraction operator $\ominus$ to support complement comparisons as mentioned in Section 4.1, if required by the utilized exceptionality measure.

Patterns are then evaluated based on their valuation bases by applying

- a model extractor χ to map valuation bases $v \in V$ onto model parameters, and
- an exceptionality measure q based on these model parameters.

These adaptations of FP-growth allow for a generic implementation. That is, the code for the main algorithm is identical for all model classes. To apply it to a new model class, only the valuation domain with its aggregation operator $\oplus$, the corresponding valuation projector ϕ, and the model extractor χ must be implemented. We call the tuple $(V, \oplus, \phi, \chi)$ a *model configuration*.

Please note, that for the very simple valuation basis, that only counts the instances, the resulting algorithm is identical to FP-growth. Furthermore, traditional subgroup discovery can be implemented in this generic algorithm by using valuation bases that count instances with a positive and a negative target concept separately, as done in the SD-Map algorithm [3]. Thus, the approach presented in this paper can be regarded as a true generalization of both, FP-growth [7] and SD-Map.

Like subgroup discovery, exceptional model mining is often applied in a top-k approach. That is, the goal is to find the best k patterns with respect to an interestingness measure. This can be accomplished by storing the current top k patterns in a separate result set and replacing its entries with higher quality patterns as required.

When using a top-k approach, substantial speed-ups can be achieved by using optimistic estimate pruning, see for example [17]. As an example, a quality function using a correlation coefficient model could exploit the fact, that the correlation coefficient never exceeds a value of 1. However, efficient boundaries need to be determined for each quality function, which may even vary for a single model class. In this paper, we focus on a generic data structure that allows for the efficient computation of model parameters. Therefore, we will not discuss possible optimistic estimate boundaries for model exceptionality measures in the context of this work.

4.3 Theorem on Condensed Valuation Bases

The approach of generalized FP-trees is especially efficient, if it is possible to derive a small condensed valuation basis for a model class. If the constructed model itself is very complex, then it seems difficult to derive suitable condensed valuation bases that are sufficient to extract the model parameters. This includes, for example, computationally expensive models that involve the learning of a bayesian network as done in [6].

Therefore, in the following we provide a characterization of model classes, for which GP-trees can be applied with strongly reduced memory requirements. We do this by drawing a parallel to data stream mining:

Theorem 1. *There is a condensed valuation domain for a given model class if and only if the following conditions are met: (1) There is a* parallel single-pass algorithm *with sublinear memory requirements to compute the model from a given set of instances which are distributed randomly on one of the (parallel) computation nodes; (2) the only communication between the computation nodes in this algorithm takes place when combining results.*

Proof. $\Rightarrow$: First, assume there is a model configuration $(V, \oplus, \phi, \chi)$ that can be used to determine the model parameters of a subgroup. We can then construct a parallel single-pass algorithm as follows: In each computation node N we loop through all assigned instances I_N, updating the respective valuation basis v_N. For each instance $i \in I_N$ we extract the valuation basis $\phi(i)$ and use it to update the current accumulated valuation basis: $v_N^{new} = v_N^{old} \oplus \phi(i)$. Thus, after each step the valuation basis v_N corresponds to all instance handled so far. After the loop through all instances of this computation node, the valuation basis v_N can be used to extract the model parameters for the set of instances I_N. Furthermore, the resulting valuation bases from different computation nodes can be combined by using the aggregation operator $\oplus$ again. This leads to a valuation basis that corresponds to all instances of the dataset. The model parameters can then be extracted using the model extractor χ; this completes the parallel single-pass algorithm for computing these parameters.

$\Leftarrow$: Assume there is a parallel single-pass algorithm with the properties presented above. Then, there is a set of variables V_C that are used in the computation within each of the nodes, which is sublinear with respect to the number of contained instances. We show, that this set of variables defines a model configuration $(V, \oplus, \phi, \chi)$. Since the algorithm is single-pass, the assignments for these variables are updated only once for each instance using the values of the model attributes for this instance. Now let v_i be the vector of values (variable assignments) of the variables V_C after the instance i is processed as the *first* instance in this computation node. Then, we can use this vector as our valuation basis projector function $\phi(i) = v_i$. This is sufficient for a valuation basis; if there was only one instance in the dataset, then a correct algorithm would be able to extract the model parameters for the model built from the single instance using only the data v_i.

Next, assume that each computational node N_j has finished the computation of its partition of the data D_j, each resulting in variable assignments v_j, which corresponds to a valuation basis. v_j must be sufficient to extract the model parameters for the data D_j, since it could be that N_j is the only computational node. The method used for this subtask can be regarded as a model extractor function χ. Now consider two valuation bases v_1 and v_2 that result from two computation nodes and are corresponding to data partitions D_1 and D_2. A correct parallel algorithm must come with an appropriate method to combine the results v_1 and v_2 into new variable assignments that is suited to extract model parameters for the data $D_1 \cup D_2$. This method can be used as a general aggregation function $\oplus$ for valuation bases. Thus, given a parallel single-pass algorithm with the properties presented above we can derive a model configuration $(V, \oplus, \phi, \chi)$. □

The proof is constructive. It describes a method to transfer parallel single-pass algorithms for specific model classes to valuation domains that can be used for efficient exceptional model mining. Some important examples of this approach are shown in the next section.

5 Valuation Bases for Important Model Classes

In this section, we discuss the application of GP-trees to different model classes. Most of the presented model classes have been proposed in [13], to which we refer for a more detailed description of the models and exceptionality measures.

Variance Model. The variance model identifies patterns, in which the variance of a single target variable X is especially high/low. Although this model features only a single model attribute, this task can not be accomplished by traditional subgroup discovery algorithms utilizing FP-trees, such as SD-Map.

For an efficient computation of the variance, we utilize the following well known formula:

$$Var(X) = E[X^2] - E[X]^2 = \frac{\sum x^2}{n} - (\frac{\sum x}{n})^2,$$

where $E[X]$ is the expected value for the variable X. For computing the variance of an attribute, only the total count, the sum of all values and the sum of all squared values are required. Formally, a model configuration $(V_\sigma, \oplus_\sigma, \phi_\sigma, \chi_\sigma)$ that is sufficient to compute the variance (or equivalently, the standard deviation) of a variable X can be defined as:

$$\begin{aligned} V_\sigma &= \mathbb{R}^3 \\ v \oplus_\sigma u &= v + u \\ \phi_\sigma(i) &= (1, i_X, {i_X}^2)^T \\ \chi_\sigma(v) &= \frac{v_3}{v_1} - (\frac{v_2}{v_1})^2 \end{aligned}$$

Each valuation basis stores a vector of three real numbers. Aggregating valuation bases using the operator $\oplus$ is equivalent to adding vectors in euclidean space. The valuation basis extracted from a single data instance i contains the constant 1 as the instance count, the value of X in i and the squared value of X in i. To extract the model parameter $Var(X)$ from a valuation basis $v \in V_\sigma$ the computation χ_σ has to be performed using the three components of the vector stored in the valuation basis v.

Correlation Model. The (Pearson product-moment) correlation coefficient $\rho(X,Y)$ measure is a very well known statistical measure that reflects the linear dependency between two numerical attributes X and Y. The correlation coefficient is defined as the fraction of the covariance and the product of the standard deviations of these two attributes: $\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$.

The covariance is defined as $Cov(X,Y) = \frac{\sum_{(x,y)}(x-\mu_X)(y-\mu_Y)}{N}$, where μ describes the mean value of the attribute and N the number of instances. We determine the measures $Cov(X,Y)$, σ_X and σ_Y independently from each other.

To efficiently compute the covariance of two variables X, Y we utilize the following pairwise update formula that was introduced in [4] for an parallel single-pass algorithm. It allows us to compute $C_S(X,Y) = Cov(X,Y) \cdot |S| = \sum_{(x,y)\in S}(x-\mu_X)(y-\mu_Y)$ for a set of data instances $S = S_1 \cup S_2, S_1 \cap S_2 = \emptyset$ given statistical information of the partitioning sets S_1 and S_2:

$$C_S(X,Y) = C_{S_1}(X,Y) + C_{S_2}(X,Y) + \frac{n_1 n_2}{n_1 + n_2}(\mu_{X,2} - \mu_{X,1})(\mu_{Y,2} - \mu_{Y,1}),$$

where, $n_1 = |S_1|$ and $n_2 = |S_2|$ denote the instance count in S_1 and S_2, and $\mu_{X,2}, \mu_{X,1}, \mu_{Y,2}, \mu_{Y,1}$ are the mean values of X and Y in the sets S_2 and S_1.

For computing the covariance for each pattern using this formula, we need to keep track of the value of C in the respective set of instances, the cardinality of the subgroup, and the mean values of the variables X and Y. The latter can be computed by the sum of the respective values and the cardinality. Therefore, using the formula above we can define a model configuration for the covariance as follows:

$$\begin{aligned}
V_{cov} &= \mathbb{R}^4 \\
v \oplus_{cov} u &= \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \oplus \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} v_1 + u_1 \\ v_2 + u_2 \\ v_3 + u_3 \\ v_4 + u_4 + \frac{v_1 u_1}{v_1 + u_1}(\frac{v_2}{v_1} - \frac{u_2}{u_1})(\frac{v_3}{v_1} - \frac{u_3}{u_1}) \end{pmatrix} \\
\phi_{cov}(i) &= (1, i_X, i_Y, 0)^T \\
\chi_{cov}(v) &= \frac{v_4}{v_1}
\end{aligned}$$

In this formalization of a model configuration the first component of a valuation basis reflects the size of the corresponding set, the second and third component store the sum of the values of X and Y and the fourth component keeps track of the measure C as defined above.

To compute the actual correlation coefficient we combine this valuation basis with the valuation basis used for the variance model in order to compute $Cov(X, Y)$, σ_X and σ_Y in a single model configuration:

$$\begin{aligned}
V_{cor} &= \mathbb{R}^6 \\
v \oplus_{cor} u &= \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix} \oplus \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} v_1 + u_1 \\ v_2 + u_2 \\ v_3 + u_3 \\ v_4 + u_4 + \frac{v_1 u_1}{v_1 + u_1}(\frac{v_2}{v_1} - \frac{u_2}{u_1})(\frac{v_3}{v_1} - \frac{u_3}{u_1}) \\ v_5 + u_5 \\ v_6 + u_6 \end{pmatrix} \\
\phi_{cor}(i) &= (1, i_X, i_Y, 0, {i_X}^2, {i_Y}^2)^T \\
\chi_{cor}(v) &= \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\frac{v_4}{v_1}}{\sqrt{\frac{v_5}{v_1} - (\frac{v_2}{v_1})^2}\sqrt{\frac{v_6}{v_1} - (\frac{v_3}{v_1})^2}} = \frac{v_1 v_4}{\sqrt{v_1 v_5 - v_2^2}\sqrt{v_1 v_6 - v_3^2}}
\end{aligned}$$

Simple Linear Regression Model. The simple linear regression model is perhaps the most intuitive statistical model to show the dependency between two numeric variables X and Y. It is built by fitting a straight line in the two dimensional space by minimizing the squared residuals e_j of the model:

$$y_j = a + bx_j + e_j$$

As proposed in [13], the difference of the slope b of this line in a subgroup and the total population (or the complement of the subgroup within the population) can be used to identify interesting patterns. As known from statistics, the slope b can be computed by the covariance of both variables and the variance of X:

$$slope(X,Y) = \frac{Cov(X,Y)}{Var(X)}$$

Thus, we define a model configuration by combining the valuation domains for the variance and the covariance, similar to the correlation model:

$$\begin{aligned}
V_{slope} &= \mathbb{R}^5 \\
v \oplus_{slope} u &= \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{pmatrix} \oplus \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \begin{pmatrix} v_1 + u_1 \\ v_2 + u_2 \\ v_3 + u_3 \\ v_4 + u_4 + \frac{v_1 u_1}{v_1+u_1}(\frac{v_2}{v_1} - \frac{u_2}{u_1})(\frac{v_3}{v_1} - \frac{u_3}{u_1}) \\ v_5 + u_5 \end{pmatrix} \\
\phi_{slope}(i) &= (1, i_X, i_Y, 0, {i_X}^2)^T \\
\chi_{slope}(v) &= \frac{Cov(X,Y)}{Var(X)} = \frac{\frac{v_4}{v_1}}{\frac{v_5}{v_1} - (\frac{v_2}{v_1})^2} = \frac{v_1 v_4}{v_1 v_5 - v_2^2}
\end{aligned}$$

Here, again the first four components represent the cardinality of the corresponding set of instances, the sum of values for X and Y and the measure C as defined above. Additionally, the last component is additionally required to compute the variance as described previously for the variance model.

Logistic Regression Model. Next, we consider the logistic regression model. This model is used for the classification of a binary target attribute $Y \in A_M$ from a set of independent binary attributes $X_j \in A_M \setminus Y, j = 1, \ldots, |A_M| - 1$. The model is given by: $y = \frac{1}{1+e-z}, z = b_0 + \sum_j b_j x_j$. An exceptional model mining goal could then be identify patterns in which the model parameters b_j differ significantly from the ones derived from the total population.

Unfortunately, to the best of the authors' knowledge until now no exact single-pass algorithm has been proposed for determining the parameters for logistic regression, due to the non-linear nature of parameter fitting. Since according to Theorem 1 the existence of such an algorithm is necessary for the existence of a sufficient condensed valuation basis, the exact computation of parameters for the logistic regression model relies on the trivial valuation basis so far.

DTM-Classifier. Next, we discuss two models based on the DTM-classifier [10]: It predicts a target attribute $Y \in A_M$ from a set of independent attributes $X_j \in A_M \setminus Y, j = 1, \ldots, |A_M| - 1$, by determining the probability of each target attribute value for each combination of values of the independent attributes. For value combinations, which did not occur in the training set, the probability distribution of the complete training set is used. Then, for a given new instance i the target value is predicted, that has the highest probability conditioned on the respective combination of values of the X_j in i. If the specific combination of the values of X_j did not occur in the training data, then the instance is classified as the most frequent target value in the complete training set. In the context of exceptional model mining, amongst others, the *Hellinger distance* has been proposed as an exceptionality measure for this model class. It measures the difference between the distribution within a subgroup S and the distribution in its complement $\bar{S}$. It is computed as

$$\sum_{y,x_1,\ldots x_k} (\sqrt{P_S(y|x_1,\ldots,x_k)} - \sqrt{P_{\bar{S}}(y|x_1,\ldots,x_k)})^2.$$

For an efficient computation in FP-trees, we store the probabilities for all value combinations of $Y, X_1, \ldots, X_{|A_M-1|}$.

In the following, we assume that all attributes are binary for the sake of simpler notation. The generalization for non-binary attributes is straightforward. Furthermore, we assume that value combinations are arranged in a predefined order, where the positive values of Y are on odd positions and the corresponding negative value of Y for the same combination of independent attribute values is immediately following. Then, a slightly simplified model configuration is given by:

$$\begin{aligned}
V_{dtm} &= \mathbb{R}^{2^{|A_M|}} \\
v \oplus_{dtm} u &= v + u \\
\phi_{dtm}(i) = (v_j) &= \begin{cases} 1, \text{ if the j-th combination of values is true in } i \\ 0, \text{ else} \end{cases} \\
\chi^{(k)}_{dtm}(v) &= \frac{v_{2k-1}}{v_{2k-1} + v_{2k}}
\end{aligned}$$

In this model configuration each component of the valuation basis corresponds to a combination of values. Please note, that in this case the valuation basis stored in each node of the GP-tree needs to store 2^k values, where k is the number of (binary) model attributes. Therefore this model configuration is not tractable for large numbers of model attributes. However, this should not be the case in most practical applications, since larger models are typically difficult to comprehend by human users.

Bayesian Networks. Bayesian networks have been proposed as complex target models for exceptional model mining [6]. Since to the best of the authors' knowledge there is currently no parallel single-pass algorithm for learning bayesian networks — which is a complex task on its own — we cannot provide an efficient valuation basis for this model class here, but refer to the trivial valuation basis. However, there is ongoing research in that area [5] which can possibly be exploited in future work.

6 Evaluation

In this section, we present runtime evaluations of the proposed approach using UCI-datasets [14] as well as a scalability study in a large real world dataset.

6.1 Runtime Evaluations on UCI Data

We evaluated the presented approach by performing runtime experiments using a set of well known UCI datasets. The algorithms were implemented in the open source data mining environment VIKAMINE[1] [2]. The experiments were performed on a standard office PC with a 2.2 GHz CPU and 2 GB RAM. As search space we used all non-model attributes in the respective dataset. Numeric attributes were discretized in five intervals using equal-frequency discretization.

In a first set of experiments we compared the runtime of the GP-tree approach for different model classes. Due to the limited space we just show the results for the exemplary *credit-g* dataset, see Table 1. In this dataset, we used the attributes *duration* and *credit_amount* as model attributes, which were discretized if necessary. We excluded these attributes from the search space for all model classes to increase the comparability of the results. Experiments on other datasets showed similar characteristics. As can be seen in the table, the runtimes differ only marginally. For the DTM-classifier, the results only differ by a small constant factor, which can explained by the more complex model configuration. The similarity of the runtimes is due to the fact that no pruning scheme is utilized; therefore, the search space is the same for all model classes.

Table 1. Runtime in seconds for the GP-growth algorithm using the credit-g dataset for different model classes and various search depth (maximum number of selectors in a single subgroup description)

Model Class	Model Attr.	2	3	4	5
Frequent Pattern	-	0.8	3.5	17.1	70.0
Subgroup Discovery	duration	0.8	3.6	17.0	69.3
Variance	duration	0.8	3.6	17.0	72.6
Linear Regression	both	0.9	3.8	18.6	77.3
Correlation Coefficient	both	0.9	3.8	18.7	77.8
DTM classifier	both	1.8	7.4	32.0	118.2

Next, we performed an extensive runtime analysis of our approach using 19 datasets from the UCI repository. For that purpose, we compared the proposed GP-growth algorithm to a simple depth first search without any specialized data structure. Due to the runtime similarity for different model classes and due to the limited space, we only show the results for one model class, that is, the slope of the linear regression. The results are shown in Table 2. It can be observed, that even at a search depth of 2 (searching only for subgroup descriptions that have a maximum of 2 selectors) the GP-growth algorithm outperforms the simple depth first search approach significantly. This difference increases for larger search depth. At a search depth of 5 GP-growth completes the task two orders of magnitude faster for all datasets. These results clearly demonstrate the power of efficient data structures such as the GP-tree.

[1] www.vikamine.org

Table 2. Runtime in seconds for different UCI-Datasets for the Linear Regression Model class for various search depth (maximum number of selectors in a single subgroup description), comparing a simple Depth-First-Search with the GP-tree

max depth	2		3		4		5	
dataset	DFS	GPG	DFS	GPG	DFS	GPG	DFS	GPG
adults	230.9	6.2	8905.7	10.4	> 6h	25.1	> 6h	65.6
autos	1.6	0.5	72.4	3.8	2279.5	21.9	> 6h	104.0
breast-w	0.2	0.1	0.9	0.1	4.5	0.1	15.8	0.1
colic	1.3	0.7	32.9	2.6	639.8	12.9	9940.8	46.4
credit-a	1.2	0.2	24.1	0.9	339.6	3.2	3849.3	9.8
credit-g	2.9	0.9	68.7	3.8	1202.0	18.6	16593.8	77.3
diabetes	0.3	0.1	2.6	0.1	16.9	0.3	87.8	0.4
forestfires	0.8	0.2	16.1	0.6	235.8	2.0	2670.2	4.3
glass	0.1	0.0	1.3	0.1	10.6	0.2	66.2	0.3
heart-h	0.2	0.1	2.7	0.1	23.9	0.3	168.9	0.6
hepatitis	0.2	0.1	3.3	0.7	40.0	3.3	373.2	12.4
housing	0.2	0.0	1.4	0.1	7.7	0.2	32.8	0.4
hypothyroid	10.4	1.4	247.7	5.1	4405.4	24.3	> 6h	131.3
ionosphere	3.1	1.9	154.7	22.5	5581.3	184.3	> 6h	1136.7
labor	0.1	0.0	1.0	0.1	12.0	0.3	108.4	0.6
segment	6.5	1.0	175.3	3.9	3437.8	16.1	> 6h	59.1
spambase	18.7	6.0	558.6	28.4	12498.8	196.1	> 6h	1869.1
vehicle	2.5	0.6	66.0	3.2	1367.3	15.0	> 6h	53.1
vowel	2.1	0.3	45.3	1.3	754.4	4.3	10064.1	10.4

6.2 Scalability Study: Social Image Data

In the following, we present a short case study on real world data that shows the advantages of the presented approach in large scale applications. As a dataset, we used publicly available metadata of pictures uploaded to the Flickr[2]-platform. More specifically, we crawled the view counts as well as all tagging information for all pictures geo-referenced to a location in Germany uploaded in 2010. We limited the dataset to tags with more than 1000 occurrences leading to a dataset of about 1.1 million instances and about 1200 tags that we used as describing attributes. Since pictures viewed by more people are naturally also tagged by more people, there is a correlation to the number of tags assigned to a picture. To evaluate the scalability of the GP-growth approach we performed the task of identifying combinations of tags (as subgroup descriptions), for which this correlation is especially strong. As a result, even for a search depth of 2, the simple DFS algorithm did not finish the task within two full days. In contrast, the same task performed by GP-growth finished in about 8 minutes.

The massive difference for this dataset can be explained by the sparseness of the tagging data, which especially favors the utilized tree structure. Furthermore, even for an increased search depth of 3, the task could be completed within

[2] www.flickr.com

10 minutes. This small difference is reasonable, as due to the sparseness of the dataset less combinations of three tags might occur in dataset than combinations of two tags. Overall, the runtime improvements for the Flickr dataset are even larger than in the previous datasets, showing the scalability of our approach.

7 Conclusions

In this paper, we have proposed a novel approach for fast exhaustive exceptional model mining: We have introduced the concept of valuation bases as an intermediate condensed data representation and presented the general GP-growth algorithm for efficient exhaustive exceptional model mining. We discussed the applicability of the proposed approach by drawing an analogy to data stream mining, and provided implementation examples for several model classes. Our runtime experiments show improvements of more than an order of magnitude in comparison to a naive exhaustive depth-first search.

For future work, we aim to analyze methods for considering the diversity of pattern discovery results, e.g., [12] in order to improve the result sets. Another interesting direction for future research is the adaptation of other more advanced data structures, such as bit vectors, for exceptional model mining.

Acknowledgment. This work has partially been supported by the VENUS research cluster at the interdisciplinary Research Center for Information System Design (ITeG) at Kassel University, and by the EU project EveryAware.

References

1. Atzmueller, M., Lemmerich, F.: Fast Subgroup Discovery for Continuous Target Concepts. In: Rauch, J., Raś, Z.W., Berka, P., Elomaa, T. (eds.) ISMIS 2009. LNCS, vol. 5722, pp. 35–44. Springer, Heidelberg (2009)
2. Atzmueller, M., Lemmerich, F.: VIKAMINE - A Rich-Client Environment for Pattern Mining and Subgroup Discovery. In: Proc. LWA 2011 (KDML Track) (2011)
3. Atzmüller, M., Puppe, F.: SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 6–17. Springer, Heidelberg (2006)
4. Bennett, J., Grout, R., Pébay, P., Roe, D., Thompson, D.: Numerically Stable, Single-Pass, Parallel Statistics Algorithms. In: IEEE International Conference on Cluster Computing and Workshops (CLUSTER 2009), pp. 1–8. IEEE (2009)
5. Bromberg, F., Patterson, B., Yaramakala, E.: Mining bayesian networks from streamed data (2003)
6. Duivesteijn, W., Knobbe, A., Feelders, A., van Leeuwen, M.: Subgroup Discovery Meets Bayesian Networks–An Exceptional Model Mining Approach. In: 10th IEEE Intl Conference on Data Mining (ICDM), pp. 158–167. IEEE (2010)
7. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns Without Candidate Generation. In: Intl. Conf. on Management of Data, pp. 1–12. ACM Press (2000)
8. Herrera, F., Carmona, C., González, P., del Jesus, M.: An Overview on Subgroup Discovery: Foundations and Applications. Knowledge and Information Systems 29(3), 495–525 (2011)

9. Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In: Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 249–271. AAAI Press (1996)
10. Kohavi, R.: The Power of Decision Tables. In: Lavrač, N., Wrobel, S. (eds.) ECML 1995. LNCS, vol. 912, pp. 174–189. Springer, Heidelberg (1995)
11. van Leeuwen, M.: Maximal Exceptions with Minimal Descriptions. Data Min. Knowl. Discov. 21(2), 259–276 (2010)
12. van Leeuwen, M., Knobbe, A.: Non-redundant Subgroup Discovery in Large and Complex Data. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 459–474. Springer, Heidelberg (2011)
13. Leman, D., Feelders, A., Knobbe, A.: Exceptional Model Mining. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 1–16. Springer, Heidelberg (2008)
14. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI Repository of Machine Learning Databases (1998), http://www.ics.uci.edu/mlearn/mlrepository.html
15. Novak, P.K., Nada Lavrac, G.I.W.: Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining. Journal of Machine Learning Research 10, 377–403 (2009)
16. Umek, L., Zupan, B.: Subgroup Discovery in Data Sets with Multi-Dimensional Responses. Intelligent Data Analysis 15(4), 533–549 (2011)
17. Wrobel, S.: An Algorithm for Multi-Relational Discovery of Subgroups. In: Komorowski, J., Żytkow, J.M. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 78–87. Springer, Heidelberg (1997)

Bidirectional Semi-supervised Learning with Graphs

Tomoharu Iwata[1] and Kevin Duh[2]

[1] NTT Communication Science Laboratories
iwata.tomoharu@lab.ntt.co.jp
[2] Nara Institute of Science and Technology
kevinduh@is.naist.jp

Abstract. We present a machine learning task, which we call bidirectional semi-supervised learning, where label-only samples are given as well as labeled and unlabeled samples. A label-only sample contains the label information of the sample but not the feature information. Then, we propose a simple and effective graph-based method for bidirectional semi-supervised learning in multi-label classification. The proposed method assumes that correlated classes are likely to have the same labels among the similar samples. First, we construct a graph that represents similarities between samples using labeled and unlabeled samples in the same way with graph-based semi-supervised methods. Second, we construct another graph using labeled and label-only samples by connecting classes that are likely to co-occur, which represents correlations between classes. Then, we estimate labels of unlabeled samples by propagating labels over these two graphs. We can find a closed-form global solution for the label propagation by using matrix algebra. We demonstrate the effectiveness of the proposed method over supervised and semi-supervised learning methods with experiments using synthetic and multi-label text data sets.

Keywords: semi-supervised learning, label propagation, multi-label classification.

1 Introduction

The performance of a classifier can be improved as the number of labeled samples is increased. However, we might not have enough labeled samples to achieve a reasonable performance because their generation incurs cost and requires time. To overcome the shortage of labeled samples, there has been great interest in methods that effectively increase training samples. For example, semi-supervised learning [1] augments training samples by using unlabeled samples. The other examples include domain adaptation [2] and class adaptation [3], where the former utilizes samples from different domains and the latter utilizes samples from different taxonomies.

In this paper, we consider a new way to improve multi-label classification performance, where we have label-only samples as well as labeled and unlabeled

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 293–306, 2012.

Table 1. Notation

Symbol	Description
$\{(\boldsymbol{x},\boldsymbol{y})\}$	labeled samples
$\{\boldsymbol{x}\}$	unlabeled samples
$\{\boldsymbol{y}\}$	label-only samples
N	number of labeled samples
U	number of unlabeled samples
O	number of label-only samples
K	number of classes
M	number of features

Table 2. Given data in supervised, semi-supervised and bidirectional semi-supervised learning

Problem	Given data
supervised	$\{(\boldsymbol{x},\boldsymbol{y})\}$
(one-directional) semi-supervised	$\{(\boldsymbol{x},\boldsymbol{y})\},\{\boldsymbol{x}\}$
bidirectional semi-supervised	$\{(\boldsymbol{x},\boldsymbol{y})\},\{\boldsymbol{x}\},\{\boldsymbol{y}\}$

samples. The labeled samples are a set of pairs of feature and label vectors $\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^{N}$, and the unlabeled samples are a set of feature vectors $\{\boldsymbol{x}_i\}_{i=N+1}^{N+U}$ without label information. The label-only samples are a set of label vectors $\{\boldsymbol{y}_i\}_{i=N+U+1}^{N+U+O}$, where corresponding feature information is unavailable. We call this setting *bidirectional semi-supervised learning*. Semi-supervised learning is defined as a task with abundant inputs (unlabeled samples) but few input-output pairs (labeled samples); so we define bidirectional semi-supervised learning as a task with abundant inputs and abundant outputs (label-only samples), but few input-output pairs. Table 1 summaries our notation, and Table 2 summaries the given data in supervised, semi-supervised and bidirectional semi-supervised learning.

This setting can be found in many real applications. For example, let us consider recommendation problems in two different domains, where the feature vector is a user's preference for items in a given domain, and the label vector is the user's binary preference for items in another domain [4, 5]. Here, we suppose that we want to estimate the preferences in the second domain. Users who have preferences in the both domains can be used for labeled samples, those who have preferences only in the first domain can be used for the unlabeled samples, and those who have preferences only in the second domain can be used for the label-only samples. Cross-lingual information retrieval [6–8] is another application, where the feature vector is a query and the label vector consists of relevant documents in a different language. Here, we might have a lot of documents in a different language for label-only samples. Other applications include automatic image annotation problems [9–11], where the feature vector is image features and the label vector is the annotations. The label-only samples can be obtained using text corpus under the assumption that correlation between annotations is related to correlation between word in the text corpus. Similarly, in multi-label text classification problems, label correlation might be available from outside label-only sources. In general, bidirectional semi-supervised problems may arise when we have disjoint datasets of features and labels.

Unlabeled samples contain information about the distribution of samples in the feature space. Semi-supervised learning methods uses this information for improving performance. Similarly, label-only samples contain information about the distribution in the label space, or information about correlations between classes. Therefore, we can expect that the multi-label classifier performance can be improved by using label-only samples.

We propose a simple and effective graph-based method for bidirectional semi-supervised learning. A number of graph-based semi-supervised learning methods, or label propagation, have been proposed [12–15] because of its simplicity and easy implementation. They have used for a wide variety of applications, such as text classification [16], image recognition [17] and protein function prediction [18]. With the graph-based semi-supervised method, a graph is constructed using labeled and unlabeled samples by connecting samples that have similar feature vectors, where each node corresponds to a sample. Then, labels are estimated by propagating labels over the constructed graph with the assumption that connected samples tend to have the same label. An advantage of graph-base methods is that we can obtain a global closed-form solution.

The proposed method is an extension of the graph-based semi-supervised methods. First, we construct a graph using labeled and unlabeled samples in the similar way with graph-based semi-supervised learning. Second, we construct another graph using labeled and label-only samples by connecting classes that are likely to co-occur, where each node corresponds to a class. Then, we estimate labels by using these two graphs with the assumption that labels of correlated classes in similar samples tend to be the same. We can obtain a global closed-form solution for the proposed method. We can use similar techniques that have been extensively studied for graph-based semi-supervised methods for the proposed method, such as techniques for constructing effective graphs and algorithms for efficient estimation.

The remainder of this paper is organized as follows. In Section 2, we formulate the proposed method, and describe a closed-form solution and an iterative estimation method. In Section 3, we briefly review related work. In Section 4, we demonstrate the effectiveness of the proposed method with experiments using synthetic and multi-label text data sets. Finally, we present concluding remarks and a discussion of future work in Section 5.

2 Proposed Method

We suppose that there are N labeled samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, U unlabeled samples $\{\boldsymbol{x}_i\}_{i=N+1}^{N+U}$, and O label-only samples $\{\boldsymbol{y}_i\}_{i=N+U+1}^{N+U+O}$. A feature vector is represented by $\boldsymbol{x}_i = (x_{im})_{m=1}^{M}$, where x_{im} is the mth element of the ith sample's

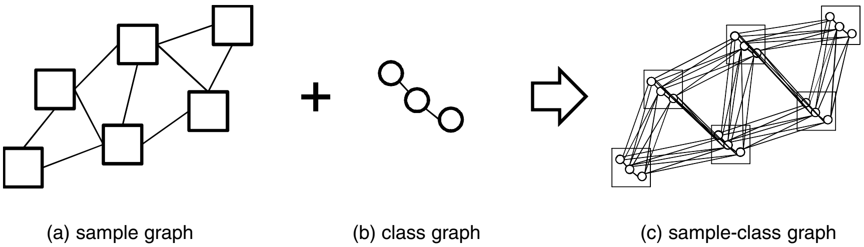


Fig. 1. Sample, class and sample-class graphs. Square and circle nodes represent sample and class, respectively.

feature vector, and M is the number of features. A label vector is represented by $\boldsymbol{y}_i = (y_{ik})_{k=1}^{K}$, where $y_{ik} = 1$ if the ith sample is categorized into class k, $y_{ik} = -1$ otherwise, $y_{ik} \in \{-1, 1\}$, and K is the number of classes. Each sample can be assigned to multiple classes. Classes that do not appear in labeled samples can appear in label-only samples. Our task is to assign labels to unlabeled samples.

First, we construct a sample graph, where nodes are the labeled and unlabeled samples. An edge between two nodes represents the similarity of feature vectors of the two samples. The edge weight can be calculated by using Gaussian kernel as follows,

$$w_{ij} = \exp\left(-\frac{\alpha}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel^2\right), \tag{1}$$

where α is the precision parameter. We can also build the sample graph with k nearest neighbors, where nodes are connected if they are k nearest neighbors in Euclidean distance, and $w_{ij} = 0$ otherwise. Figure 1 (a) shows an example of a sample graph, where a square node represents a sample.

Second, we construct a class graph, where nodes are the classes that appear in the labeled and label-only samples. An edge between two nodes represents the similarity of the two classes, or how likely the two classes co-occur. The edge weight can be calculated by using Gaussian kernel in the similar way to the sample graph,

$$v_{kl} = \exp\left(-\frac{\beta}{2} \parallel \boldsymbol{y}^{(k)} - \boldsymbol{y}^{(l)} \parallel^2\right), \tag{2}$$

where β is the precision parameter, and $\boldsymbol{y}^{(k)} = (y_{1k}, \cdots, y_{Nk}, y_{N+U+1,k}, \cdots, y_{N+U+O,k})$ is an $N + O$ dimensional vector that consists of the kth elements of label vectors in the labeled and label-only samples. $\boldsymbol{y}^{(k)}$ can be used with $L2$ normalization so that the weights correlate to their cosine similarities. Figure 1 (b) shows an example of a class graph, where a circle node represents a class. Note that there are $N + U$ nodes in the sample graph, and K nodes in the class graph.

Then, we estimate labels for unlabeled samples using the sample and class graphs. We suppose that f_{ik} is a real valued relaxation of y_{ik}, which is to be estimated. We assume that labels of correlated classes (which are connected in the class graph) in similar samples (which are connected in the sample graph) are likely to be similar. This can be achieved by minimizing the following function,

$$E = \frac{1}{2} \sum_{i,j=1}^{N+U} \sum_{k,l=1}^{K} w_{ij} v_{kl} (f_{ik} - f_{jl})^2, \tag{3}$$

with the constraint of $f_{ik} = y_{ik}$ on the labeled data $i = 1, \cdots, N$. When w_{ij} is high (feature vectors of i and j are similar) and v_{kl} is high (classes k and l are correlated), the estimated value for class k of the ith sample needs to be similar to that for class l of the jth sample so as to minimize the objective function E. Therefore, by minimizing the objective function, we can find estimated labels, where correlated classes in similar samples have similar labels.

The proposed method can be seen as label propagation on a sample-class graph that is build by combining sample and class graphs as shown in Figure 1 (c), where the correlated classes in the similar samples are connected. In the sample-class graph, each node corresponds to a class of each sample, and the number of nodes is $(N + U)K$.

With graph-based semi-supervised learning methods, the following objective function is minimized,

$$E = \frac{1}{2} \sum_{i,j=1}^{N+U} w_{ij} \sum_{k=1}^{K} (f_{ik} - f_{jk})^2, \tag{4}$$

where they assume that similar samples have similar labels. However, they do not consider the correlation between classes. The graph-based semi-supervised learning methods can be seen as label propagation on a sample graph without class graphs. The proposed method with $v_{kl} = 1$ if $k = l$ and $v_{kl} = 0$ otherwise coincides with the graph-based semi-supervised method.

2.1 Closed-Form Solution

We can find a closed-form global solution for the minimization of the objective function E by using matrix algebra. The proposed method propagates labels on the sample-class graph as shown in Figure 1. Therefore, we can use the same algorithm for finding the solution with label propagation for semi-supervised learning [15]. Let $\boldsymbol{A}$ be an $(N+U)K \times (N+U)K$ matrix, whose $(iK+k, jK+l)$th element is $A_{iK+k,jK+l} = w_{ij} v_{kl}$ as follows,

$$\boldsymbol{A} = \begin{pmatrix} w_{11}v_{11} & w_{11}v_{12} \cdots & w_{11}v_{1K} \\ w_{11}v_{21} & w_{11}v_{22} \cdots & \vdots \\ \vdots & \vdots \quad \ddots & \vdots \\ w_{N+U,1}v_{K1} & \cdots \quad \cdots & w_{N+U,N+U}v_{KK} \end{pmatrix}, \tag{5}$$

which represents the sample-class graph. Let $\boldsymbol{D}$ be a diagonal matrix, whose ith diagonal element is

$$D_{ii} = \sum_{j=1}^{(N+U)K} A_{ij}. \tag{6}$$

The graph Laplacian matrix $\boldsymbol{L}$ is defined as,

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}. \tag{7}$$

Let $\boldsymbol{f} = (f_{11}, f_{12}, \cdots, f_{1K}, f_{21}, \cdots, f_{N+U,K})^\top$ be the vector of f values on all of the labeled and unlabeled samples. The objective function E can be written as,

$$E = \boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f}. \tag{8}$$

Then, we can obtain the following closed-form solution by solving the constrained optimization problem using Lagrange multipliers,

$$\boldsymbol{f}_l = \boldsymbol{y}_l, \tag{9}$$

$$\boldsymbol{f}_u = -\boldsymbol{L}_{uu}^{-1} \boldsymbol{L}_{ul} \boldsymbol{y}_l. \tag{10}$$

Here, $\boldsymbol{f} = (\boldsymbol{f}_l, \boldsymbol{f}_u)$ and

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{L}_{ll} & \boldsymbol{L}_{lu} \\ \boldsymbol{L}_{ul} & \boldsymbol{L}_{uu} \end{pmatrix}, \tag{11}$$

which are partitioned with respect to values of labeled and unlabeled samples.

2.2 Iterative Algorithm

We can also obtain a solution by using an iterative algorithm [19, 20]. When the graphs are sparse, the iterative algorithm is efficient and reduces required memory. Sparse graphs can be obtained by using k nearest neighbor graph construction for sample and class graphs. With the closed-form solution, we need the inverse of a $UK \times UK$ matrix $\boldsymbol{L}_{uu}$, whose inverse is not sparse even if $\boldsymbol{L}_{uu}$ is sparse. The numbers of unlabeled samples U and classes K might be large, and it might require a huge memory space for storing the $UK \times UK$ dense matrix. On the other hand, the iterative algorithm does not need to calculate any dense matrices. First, we initialize estimates as follows,

$$\boldsymbol{f}_l^{(0)} \leftarrow \boldsymbol{y}_l, \tag{12}$$

$$\boldsymbol{f}_u^{(0)} \leftarrow (0, 0, \cdots, 0). \tag{13}$$

Then, we iterate the following updates until convergence,

$$\boldsymbol{f}^{(t+1)} \leftarrow \boldsymbol{D}^{-1} \boldsymbol{A} \boldsymbol{f}^{(t)}, \tag{14}$$

$$\boldsymbol{f}_l^{(t+1)} \leftarrow \boldsymbol{y}_l, \tag{15}$$

where $\boldsymbol{f}^{(t)}$ is the estimates of the tth iteration. We can find the unique fixed point by the iterative algorithm.

2.3 Induction

The method described before is transductive, which means that the method estimates labels of the given unlabeled samples. When we newly obtain unlabeled samples to be estimated, we can estimate their labels by combining previously given samples and the newly obtained samples. However, it is inefficient. We can efficiently estimate labels for out of samples using the estimation result for the previously given samples as follows,

$$f_{ik} = \frac{\sum_{j=1}^{N+U} \sum_{l=1}^{K} w_{ij} v_{kl} f_{jl}}{\sum_{j=1}^{N+U} \sum_{l=1}^{K} w_{ij} v_{kl}}, \tag{16}$$

where f_{jl} is the estimated labels of the previously given samples.

3 Related Work

Bidirectional semi-supervised learning is a new type of machine learning task, and the proposed method is a simple method based on graphs. A lot of graph-based semi-supervised learning methods have been proposed. However, most of them are for single-label classification. Those methods can be applied to multi-label classification by estimating each label independently. However, by considering the class interdependence for multi-label classification, the performance can be improved and some those types of graph-based semi-supervised learning methods have been proposed [21–23, 4]. For example, [22] proposed a method that estimates labels so that multiple labels for each sample satisfy the given correlations between classes. Other multi-label classifiers also uses class correlation [24]. However, they utilize only classes that appeared in the labeled samples. On the other hand, the proposed method can utilize classes that do not appear in the labeled samples by representing class correlation by a graph, and propagate labels over the graph. With the proposed method, even if two classes do not directly co-occur, label information can propagate through edges. In [21], correlation between labels for each sample is considered. In contrast, the proposed method considers correlation between labels not only in one sample but also in multiple similar samples.

4 Experiments

4.1 Data

We demonstrate the effectiveness of the proposed method using the following two data sets: Swissroll and Patent.

The Swissroll data [25, 26] are synthetic, where samples of three dimensional feature vectors are lying on a two dimensional nonlinear manifold as shown in Figure 2. We augmented the swissroll data set with multiple labels. We generated label vectors so that they became similar if their feature vectors were located

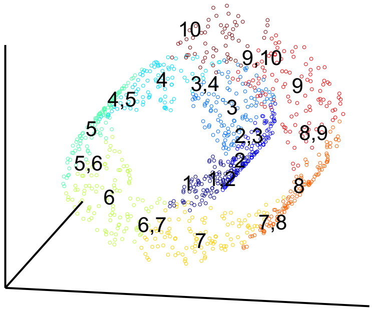


Fig. 2. Swissroll data. Each point represents a feature vector and its color represents the labels. The numbers show that nearby samples are assigned to those classes.

closely in the two dimensional nonlinear manifold, where we set the number of classes at $K = 10$. We generated samples, where the number of labeled, unlabeled and label-only samples were $N = 10$, $U = 1,000$ and $O = 1,000$ respectively.

The Patent data consist of patents published in Japan from January to March in 2004, to which International Patent Classification (IPC) codes were attached by experts according to their content. We used bag-of-words of a patent for the feature vector, where the number of words was $M = 104,621$, and we normalized each feature vector by $L2$ norm. We used the most frequently occurred 500 IPC codes in the corpus for the classes, $K = 500$. We sampled 10 labeled samples, $1,000$ unlabeled samples, and $5,000$ label-only samples from the corpus.

Figure 3 shows a class graph of the Patent data set. Here, each node represents a class, and it is visualized by [27] so that connected nodes are located closely. Some classes form clusters, and we can see the structure of classes from the visualization result. In the Patent data set, there are correlated classes, such as 'transmitter' and 'receiver', 'distinct material semiconductor' and 'distinct alignment semiconductor', and 'system to generate signals for adjusting focus' and 'automatic focus adjusting system'.

4.2 Measurements

For the evaluation measurements, we used mean reciprocal rank (MRR) and normalized discounted cumulative gain (NDCG) [28], which were widely used in evaluating ranking problems. We used ranking measurements because they give higher scores when true classes are ranked higher than false classes even if the estimated classes did not exactly match with the true classes. They were also used for multi-label classification [29–31].

The MRR is the average of the reciprocal ranks, and the MRR of the ith sample is given as follows,

$$MRR_i = \frac{1}{|\boldsymbol{y}_i|} \sum_{k:y_{ik}=1} \frac{1}{\mathrm{rank}_{ik}}, \tag{17}$$

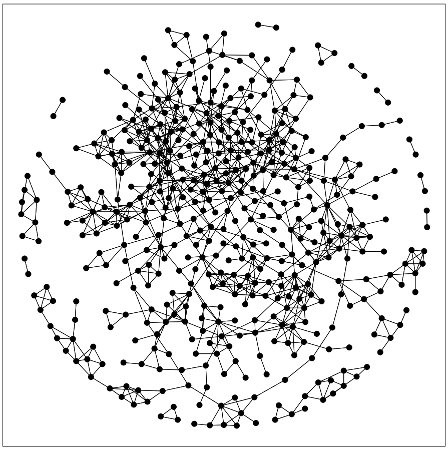


Fig. 3. A class graph of the Patent data set, where each node represents a class or IPC code

where rank_{ik} is the rank of class k of the ith sample in the estimated result, and $|\boldsymbol{y}_i|$ represents the number of classes that satisfy $y_{ik} = 1$.

The DCG is calculated as follows,

$$DCG_i = g_{i1} + \sum_{k=2}^{K} \frac{g_{ik}}{\log_2 k}, \tag{18}$$

where $g_{ik} = 1$ if the kth ranked estimated class is the true class for the ith sample, and $g_{ik} = 0$ otherwise. NDCG can be obtained by normalizing DCG by the maximum possible DCG, and NDCG lies on the interval 0.0 to 1.0. With the proposed method, classes were ranked according to values f_{ik} for each unlabeled sample. Higher MRR and NDCG represent better classification performance.

4.3 Compared Methods

We compared the proposed method, which uses all of the labeled, unlabeled and label-only samples, with a supervised method, which uses only labeled samples, and a semi-supervised method, which uses labeled and unlabeled samples.

For the supervised method (SL), we used a maximum entropy model, which is a discriminative classifier, and it has achieved high performance for text classification [32]. The maximum entropy model estimates the probability distribution that maximizes entropy under the constraints imposed by the given data. The probability that the ith sample is classified into class k is calculated as follows,

$$P(k|i) = \frac{\exp(\boldsymbol{\theta}_k^\top \boldsymbol{x}_i)}{\sum_{l=1}^{K} \exp(\boldsymbol{\theta}_l^\top \boldsymbol{x}_i)}, \tag{19}$$

Table 3. Average MRR and NDCG in the Swissroll data set and their standard deviation

	MRR	NDCG
SL	0.520 ± 0.049	0.738 ± 0.047
SSL	0.602 ± 0.048	0.814 ± 0.053
Proposed	$\mathbf{0.675} \pm \mathbf{0.047}$	$\mathbf{0.900} \pm \mathbf{0.039}$

Table 4. Average MRR and NDCG in the Patent data set and their standard deviation

	MRR	NDCG
SL	0.029 ± 0.021	0.157 ± 0.021
SSL	0.025 ± 0.018	0.153 ± 0.019
Proposed	$\mathbf{0.034} \pm \mathbf{0.023}$	$\mathbf{0.166} \pm \mathbf{0.026}$

where $\boldsymbol{\theta}_k$ is a parameter vector for class k. The labels were ranked according this estimated probability. The parameters can be obtained using maximum a posteriori (MAP) estimation with Gaussian priors. We chose the hyper-parameters for the Gaussian priors from $\{10^{-2}, 10^{-1}, 1\}$ that achieved the best performance.

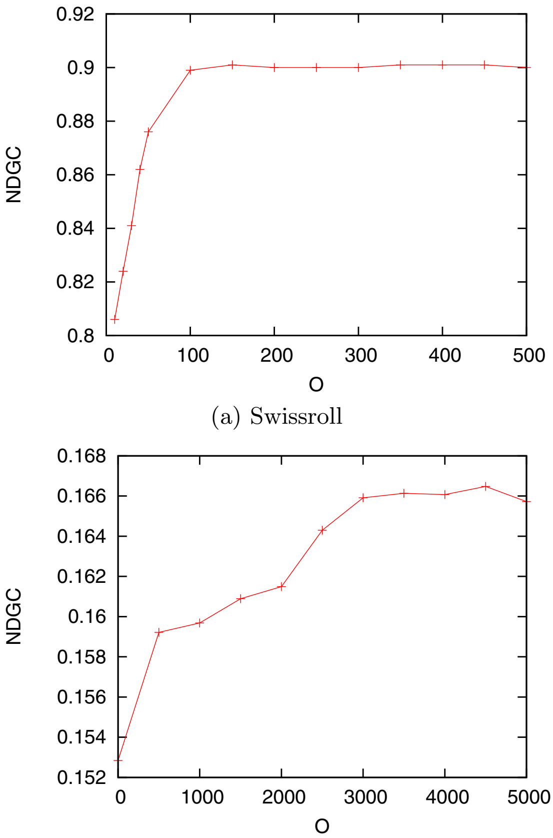
For the semi-supervised method (SSL), we used a graph-based semi-supervised method, or label propagation [12]. The graph-based semi-supervised method coincides with the proposed method when the class graph is constructed with $v_{kl} = 1$ if $k = l$, and $v_{kl} = 0$ otherwise as described before. We set the precision parameter for Gaussian kernel at $\alpha = 1$, and the number of neighbors at 10 when we construct the graph.

With the proposed method, we set the precision parameters for Gaussian kernel at $\alpha = 1$ and $\beta = 1$, and the number of neighbors at 10 when we construct both of the sample and class graphs. With the semi-supervised and proposed method, we estimated labels using iterative algorithms.

4.4 Results

Tables 3 and 4 show the averages of MRR and NDCG and their standard deviations over 100 experiments with Swissroll and Patent data sets, respectively. The proposed method achieved the best performance in both data sets. This result indicates that the proposed method can appropriately assign labels through its use of label-only samples as well as unlabeled samples.

Figure 4 shows NDCGs achieved by the proposed method with different numbers of label-only samples in Swissroll and Patent data sets. As the number of label-only samples increases, the NDCG increases. The MRR showed the same tendency of the NDCG. This result implies that the proposed method can obtain relationships between classes more precisely by using more label-only samples.



(a) Swissroll

(b) Patent

Fig. 4. NDCG with different numbers of label-only samples

5 Conclusion

We presented bidirectional semi-supervised learning, which is a novel machine learning task to improve performance by using label-only samples as well as labeled and unlabeled samples. We then proposed a simple and effective graph-based method for bidirectional semi-supervised learning. The proposed method assumes that correlated classes are likely to have the same labels among the similar samples. The correlated classes can be found by using labeled and label-only samples, and the similar samples can be found by using labeled and unlabeled samples. In experiments with synthetic and text data sets, we confirmed that the proposed method can improve the performance of multi-label classification.

Although our results have been encouraging as a first step towards bidirectional semi-supervised learning, we must extend our approach in a number of directions. First, we can extend the proposed method by using more advanced techniques for graph-based semi-supervised learning because the proposed method uses the same framework with the graph-based semi-supervised learning methods. Examples include methods for learning weight matrices [12], and efficient algorithms for label estimation [33, 34].

Second, we need to investigate methods for bidirectional semi-supervised learning other than the proposed graph-based method. In the semi-supervised learning, a wide variety of methods have been proposed such as transductive SVMs [35, 36], methods using generative models [1, 37] as well as graph-based methods. These methods might be helpful for considering new bidirectional semi-supervised learning methods.

Finally, we would like to evaluate the proposed method in other real applications. Application examples include collaborative filtering [4, 5], cross-lingual information retrieval [6–8], and image annotation [10, 11].

References

1. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.M.: Text classification from labeled and unlabeled documents using EM. Machine Learning 39(2/3), 103–134 (2000)
2. Daume III, H., Marcu, D.: Domain adaptation for statistical classifiers. Journal of Artificial Intelligence Research 26, 101–126 (2006)
3. Iwata, T., Tanaka, T., Yamada, T., Ueda, N.: Improving classifier performance using data with different taxonomies. IEEE Transactions on Knowledge and Data Engineering 23(11), 1668–1677 (2011)
4. Li, T., Yan, S., Kweon, T.M.I.S.: Local-driven semi-supervised learning with multi-label. In: IEEE International Conference on Multimedia and Expo., ICME 2009, pp. 1508–1511 (2009)
5. Nakatsuji, M., Fujiwara, Y., Tanaka, A., Uchiyama, T., Ishida, T.: Recommendations over domain specific user graphs. In: Proceeding of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, pp. 607–612 (2010)
6. Dumais, S.T., Landauer, T.K., Littman, M.L.: Automatic cross-linguistic information retrieval using latent semantic indexing. In: Proceedings of Workshop on Cross-Linguistic Information Retrieval in SIGIR 1996, pp. 16–23 (1996)
7. Xu, J., Weischedel, R., Nguyen, C.: Evaluating a probabilistic model for cross-lingual information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2001, pp. 105–110 (2001)
8. Platt, J.C., Toutanova, K., Yih, W.: Translingual document representations from discriminative projections. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, pp. 251–261 (2010)
9. Blei, D.M., Jordan, M.I.: Modeling annotated data. In: SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 127–134 (2003)
10. Socher, R., Fei-Fei, L.: Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 966–973 (2010)

11. Kimura, A., Kameoka, H., Sugiyama, M., Nakano, T., Maeda, E., Sakano, H., Ishiguro, K.: SemiCCA: Efficient semi-supervised learning of canonical correlations. In: Proceedings of IAPR International Conference on Pattern Recognition, ICPR 2010, pp. 2933–2936 (2010)
12. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the 20th International Conference on Machine Learning, ICML 2003, pp. 912–919 (2003)
13. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Scholkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems 16, pp. 321–328. MIT Press (2004)
14. Belkin, M., Matveeva, I., Niyogi, P.: Regularization and Semi-supervised Learning on Large Graphs. In: Shawe-Taylor, J., Singer, Y. (eds.) COLT 2004. LNCS (LNAI), vol. 3120, pp. 624–638. Springer, Heidelberg (2004)
15. Zhu, X., Goldberg, A.B.: Introduction to Semi-Supervised Learning. In: Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool (2009)
16. Subramanya, A., Bilmes, J.: Soft-supervised learning for text classification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, pp. 1090–1099. Association for Computational Linguistics, Stroudsburg (2008)
17. Cao, L., Luo, J., Huang, T.S.: Annotating photo collections by label propagation according to multiple similarity cues. In: Proceedings of the 16th ACM International Conference on Multimedia, MM 2008, pp. 121–130. ACM, New York (2008)
18. Kato, T., Kashima, H., Sugiyama, M.: Robust label propagation on multiple networks. IEEE Transactions on Neural Networks 20(1), 35–44 (2009)
19. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University (2002)
20. Bengio, Y., Delalleau, O., Roux, N.L.: Label propagation and quadratic criterion. In: Chapelle, O., et al. (eds.) Semi-Supervised Learning, MIT Press, Cambridge (2006)
21. Wang, J., Zhao, Y., Wu, X., Hua, X.S.: Transductive multi-label learning for video concept detection. In: Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval, MIR 2008, pp. 298–304. ACM, New York (2008)
22. Zha, Z.J., Mei, T., Wang, J., Wang, Z., Hua, X.S.: Graph-based semi-supervised learning with multiple labels. J. Vis. Comun. Image Represent. 20, 97–103 (2009)
23. Liu, W., Chang, S.F.: Robust multi-class transductive learning with graphs. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 381–388 (2009)
24. Liu, Y., Jin, R., Yang, L.: Semi-supervised multi-label learning by constrained nonnegative matrix factorization. In: Proceedings of the 21st National Conference on Artificial Intelligence, AAAI 2006, pp. 421–426. AAAI Press (2006)
25. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
26. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000)
27. Matsubayashi, T., Yamada, T.: The hierarchical individual timestep method for large-scale graph drawing. In: Proc. of the 15th International Symposium on Graph Drawing, GD 2007 (2007)
28. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. 20, 422–446 (2002)

29. Lin, X., Chen, X.W.: Mr.KNN: soft relevance for multi-label classification. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010, pp. 349–358. ACM, New York (2010)
30. Angelova, R., Kasneci, G., Weikum, G.: Graffiti: graph-based classification in heterogeneous networks. In: World Wide Web (2011)
31. Yang, Y., Gopal, S.: Multilabel classification with meta-level features in a learning-to-rank framework. Machine Learning (2011)
32. Nigam, K., Lafferty, J., McCallum, A.: Using maximum entropy for text classification. In: Proceedings of IJCAI 1999 Workshop on Machine Learning for Informat ion Filtering, pp. 61–67 (1999)
33. Garcke, J., Griebel, M.: Semi-supervised learning with sparse grids. In: Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data (2005)
34. Mahdaviani, M., Freitas, O.D., Fraser, B., Hamze, F.: Fast computational methods for visually guided robots. In: Proc. of the International Conference on Robotics and Automation, ICRA 2005 (2005)
35. Bennett, K.P., Demiriz, A.: Semi-supervised support vector machines. In: Advances in Neural Information Processing Systems, pp. 368–374. MIT Press (1999)
36. Fung, G., Mangasarian, O.L.: Semi-supervised support vector machines for unlabeled data classification. Optimization Methods and Software 15, 29–44 (2001)
37. Fujino, A., Ueda, N., Saito, K.: Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(3), 424–437 (2008)

Coupled Bayesian Sets Algorithm for Semi-supervised Learning and Information Extraction

Saurabh Verma[1] and Estevam R. Hruschka Jr.[2]

[1] Institute of Technology Banaras Hindu University
Varanasi-India
saurabh.verma.ece08@itbhu.ac.in
[2] Federal University of Sao Carlos
SP-Brazil
estevam@dc.ufscar.br

Abstract. Our inspiration comes from Nell (Never Ending Language Learning), a computer program running at Carnegie Mellon University to extract structured information from unstructured web pages. We consider the problem of semi-supervised learning approach to extract category instances (e.g. country(USA), city(New York)) from web pages, starting with a handful of labeled training examples of each category or relation, plus hundreds of millions of unlabeled web documents. Semi-supervised approaches using a small number of labeled examples together with many unlabeled examples are often unreliable as they frequently produce an internally consistent, but nevertheless, incorrect set of extractions. We believe that this problem can be overcome by simultaneously learning independent classifiers in a new approach named Coupled Bayesian Sets algorithm, based on Bayesian Sets, for many different categories and relations (in the presence of an ontology defining constraints that couple the training of these classifiers). Experimental results show that simultaneously learning a coupled collection of classifiers for random 11 categories resulted in much more accurate extractions than training classifiers through original Bayesian Sets algorithm, Naive Bayes, BaS-all and Coupled Pattern Learner (the category extractor used in NELL).

Keywords: Semi supervised learning, information extraction.

1 Introduction

The Web can be seen as a powerful source of knowledge. Translating the Web content into a structured knowledge base containing facts about entities (e.g., **Company**(*Disney*)) and also about semantic relations between those entities (e.g. **CompanyIndustry**(*Disney*, *entertainment*)) would be of great use to many applications. Machine learning approaches have been successfully employed in tasks such as information extraction from text, where the main goal is to learn to extract instances of various categories of entities (e.g., **Athlete**(*Carl*

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 307–322, 2012.

Lewis), **City**(*Pittsburgh*), **Company**(*Google Inc.*), etc.) as well as instances of semantic relations (e.g., **CompanyLocatedInCity**(*Google Inc.*,*Pittsburgh*)) from structured and unstructured text [1–3].

One of the drawbacks of *entity* and *semantic relation* instance extractors based on supervised learning approaches is that they tend to be costly. Traditionally, such an approach requires a substantial number of labeled training examples for each target category and semantic relation. Therefore, many researchers have explored semi-supervised learning methods that use only a small number of labeled examples, along with a large volume of unlabeled text [4]. While such semi-supervised learning methods are promising, they might exhibit low accuracy, mainly, because the limited number of initial labeled examples tends to be insufficient to reliably constrain the learning process, thus, raising concept drift problems [5, 6].

The Bayesian Sets algorithm, proposed in [7], was designed to extract *entity* instances using a few labeled examples and a number of unlabeled examples (in a task traditionally known as *set expansion*). It can be considered a Bayesian inference method that, when applied to exponential family models with conjugate priors, can be implemented using exact algorithms that tend to be computationally efficient. Recent studies [8, 9] have shown, however, that the direct application of Bayesian Sets may produce poor results in tasks such as information extraction from text. In addition, when Bayesian Sets are applied to problems in which the number of labeled examples is too small, the induced results tend to be deteriorated.

NELL[1] (Never-Ending Language Learner) [10] is a computer system that runs 24 hours per day, 7 days per week. It was started up on January, 12th, 2010 and should be running forever, gathering more and more facts from the web to populate its own knowledge base. In a nutshell, NELL's knowledge base (KB) is an ontology defining hundreds of categories and semantic relations that should be populated by the system. One of the main components of NELL is called CPL, which is described in more details in [4] and works as a free-text knowledge extractor which learns and uses the learned category and semantic relation contextual patterns (e.g. "*mayor of X*" and "*X plays for Y*"), to extract instances of each category and each semantic relation defined in the KB.

The hypothesis explored in this paper is that we can follow the ideas proposed in [10, 9], and achieve much higher accuracy in semi-supervised learning by coupling the simultaneous training of many extractors using a *Coupled Bayesian Sets* (CBS) algorithm to help NELL populating its own KB with more precision than the current CPL component. The intuition here is that the under-constrained semi-supervised learning task can be made easier by adding new constraints that arise from coupling the training of many extractors based on Bayesian Sets. Following NELL's principles, we present an approach in which the input to the semi-supervised learner is an ontology defining a set of target

[1] `http://rtw.ml.cmu.edu`

categories and semantic relations, a handful of seed examples for each category and for each semantic relation, and also, a set of constraints that couple the various categories and relations (e.g., Person and Sport are mutually exclusive). We show that, given this input and a huge set of unlabeled data (text from Web pages written in English), and using a semi-supervised learning approach, CBS can achieve very significant accuracy improvements by coupling the training of extractors for dozens of categories and relations. In addition, CBS allows the system to automatically identify new constraints suggesting new instances that can be considered as mutually exclusive for a specific category.

2 Related Work

The literature shows that bootstrapping approaches used to information extraction can yield impressive results with little initial human effort (in labeling examples). Bootstrapping approaches [11–13] start with a small number of labeled seed examples and iteratively grow the set of labeled examples using high-confidence labels from the current model. Such approaches have shown promising results in applications such as web page classification, named entity classification, parsing, and machine translation, among others. After many iterations, however, accuracy typically declines mainly because errors in labeling tend to accumulate, a problem that has been referred to as semantic drift. To reduce errors introduced in under-constrained semi-supervised learning, several methods have been considered. Coupling the learning of category extractors by using positive examples of one category as negative examples for others has been shown to help limiting such a decline in accuracy[4]. Also, entity set expansion using topic information can alleviate semantic drift in bootstrapping entity set expansion [8].

Bayesian Sets (BS) algorithm is the basis for Coupled Bayesian Sets (CBS) presented in Section 3 of this paper. BS was proposed in [7], and its main idea is to take a query consisting of a small set of items (labeled examples), and, based on that query, the algorithm returns additional items (from a set of unlabeled examples) which belong in this set. It computes a score for each item by comparing the posterior probability of that item given the set, to the prior probability of the item itself. These probabilities are computed with respect to a statistical model for the data, and since the parameters of this model are unknown they are marginalized out. As proposed in [7], let D be a dataset of items, and x be an item from this set. Consider also that the user provides a query set D_c which is a small subset of D. Then, Bayesian Sets computes the ratio:

$$score(x) = \frac{p(x|D_c)}{p(x)} = \frac{p(x, D_c)}{p(x)} \qquad (1)$$

which can be interpreted as the ratio of the joint probability of observing x and D_c, to the probability of independently observing x and D_c. Intuitively, this ratio compares the probability that x and D_c were generated by the same model with the same (though unknown) parameters θ, to the probability that x and D_c came from models with different parameters θ and θ'.

The Bayesian Sets approach is based on an unsupervised idea of clustering together items that belong to the same set. Thus, it defines a cluster assuming that the data points in the cluster all come independently and identically distributed from some simple parameterized statistical model. To have a better understanding on that, consider for example, that the parameterized model is $p(x|\theta)$, where θ are the parameters. In this example, if the all data points in D_c belong to one cluster, then under this definition they were generated from the same setting of the parameters. The problem in this assumption is that the parameters setting is unknown, thus, BS averages over all possible parameter values weighted by some prior density on parameter values, $p(\theta)$. Following along these lines it is possible to estimate probabilities on x, D_c and θ as in equations (2) ,(3), (4) and (5):

$$p(x) = \int p(x|\theta)p(\theta)\mathrm{d}\theta \tag{2}$$

$$p(D_c) = \int \prod_{x_i \in D_c} p(x_i|\theta)p(\theta)\mathrm{d}\theta \tag{3}$$

$$p(x|D_c) = \int p(x|\theta)p(\theta|D_c)\mathrm{d}\theta \tag{4}$$

$$p(\theta|\ D_c) = \frac{p(D_c|\theta)p(\theta)}{p(D_c)} \tag{5}$$

Considering that there are many more items (in the set of unlabeled examples) that are not members of a target set $\mathcal{T}$ than items that are members of $\mathcal{T}$, the data can be considered binary and sparse. Thus, it is possible to have log of the score linear in x. Therefore, still following [7], let's assume each item $x_i \in D_c$ is a binary vector $x_i = (x_{i1}, ..., x_{iJ})$ where $x_{ij} \in \{0, 1\}$, and that each element of x_i has an independent Bernoulli distribution as in equation (6):

$$p(x_i|\theta) = \prod_{i=1}^{J} \theta_j^{x_{ij}} (1 - \theta_j)^{1 - x_{ij}} \tag{6}$$

It is well-known that the conjugate prior for the parameters of a Bernoulli distribution is the Beta distribution:

$$p(\theta|\alpha, \beta) = \prod_{j=1}^{J} \frac{\mathit{\Gamma}(\alpha_j + \beta_j)}{\mathit{\Gamma}(\alpha_j)\mathit{\Gamma}(\beta_j)} \theta_j^{\alpha_j - 1} (1 - \theta_j)^{\beta_j - 1} \tag{7}$$

where α and β are hyperparameters, and the Gamma function (Γ) is a generalization of the factorial function. For a query $D_c = x_i$ consisting of N vectors it is easy to show that:

$$p(D_c|\alpha,\beta) = \prod_j \frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j)\Gamma(\beta_j)} \frac{\Gamma(\tilde{\alpha_j})\Gamma(\tilde{\beta_j})}{\Gamma(\tilde{\alpha_j}+\tilde{\beta_j})} \tag{8}$$

where $\tilde{\alpha} = \alpha + \sum_{i=1}^{N} x_{ij}$ and $\tilde{\beta} = \beta + N - \sum_{i=1}^{N} x_{ij}$. For an item $x = (x_{.1} \ldots x_{.J})$ the score, written with the hyperparameters explicit, can be computed as follows:

$$\begin{aligned} score(x) &= \frac{p(x|D_c,\alpha,\beta)}{p(x|\alpha,\beta)} \\ &= \prod_j \frac{\frac{\Gamma(\alpha_j+\beta_j+N)}{\Gamma(\alpha_j+\beta_j+N+1)} \frac{\Gamma(\tilde{\alpha_j}+x_{.j})\Gamma(\tilde{\beta_j}+1-x_{.j})}{\Gamma(\tilde{\alpha_j})\Gamma(\tilde{\beta_j})}}{\frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j+\beta_j+1)} \frac{\Gamma(\alpha_j+x_{.j})\Gamma(\beta_j+1-x_{.j})}{\Gamma(\alpha_j)\Gamma(\beta_j)}} \end{aligned} \tag{9}$$

The log of the score is linear in x:

$$\log score(x) = c + \sum_j q_j x_{.j} \tag{10}$$

where

$$c = \sum_j \log(\alpha_j + \beta_j) - \log(\alpha_j + \beta_j + N) + \log \tilde{\beta}_j - \log \beta_j \tag{11}$$

and

$$q_j = \log \tilde{\alpha_j} - \log \alpha_j - \log \tilde{\beta}_j + \log \beta_j \tag{12}$$

One of the most important assumptions to make Bayesian Sets a very fast method, in practice, is that if the entire data set D is stored into one large matrix X with J columns, it is possible to compute the vector s of log scores for all points using a single matrix vector multiplication

$$s = c + Xq \tag{13}$$

Thus, for sparse data sets this linear operation can be implemented very efficiently. Each query D_c corresponds to computing the vector q and scalar c. As aforementioned, the set of unlabeled examples tend to be sparse in a set expansion task. In addition, as pointed out in [7], this can also be done efficiently if the query is also sparse, since most elements of q will equal $\log \beta_j - \log(\beta + N)$ which is independent of the query.

In [9], the Bayesian Sets weakness (that can be observed when it is applied to a problem having too few initially labeled examples) is investigated based on an *Iterative Bayesian Sets* proposal. It explores the fact the seed data mean must be greater than the instance data mean on feature j. Only such kind of features can be regarded as high-quality features in Bayesian Sets. Unfortunately, it is not always the case due to the idiosyncrasy of the data. There are many high-quality features, whose seed data mean may be even less than the candidate data mean.

In that proposed approach, however, the *Iterative Bayesian Sets* still leaves room to the insertion of new constraints (which are not explored) to adjust the problem in a way that wrong extractions can be filtered out from the self-labeling results. These new constraints are explored in our proposed approach described in Section 3.2.

When considering related works focusing on *Machine Reading*, there are interesting approaches that do not implement the never-ending learning idea (as done in NELL). The *KnowItAll* system, by Etizioni and co-workers [14] and its extensions [15, 16], also, the *Yago* system [17] are good examples, although they implement different strategies on how to build a system that can read the Web.

3 Coupled Bayesian Sets - CBS

This section describes the idea of coupling semi-supervised learning of multiple functions to constrain Bayesian Sets. Our *Coupled Bayesian Sets* method starts by training classifiers based on a small amount of labeled data, then uses these classifiers to label unlabeled data. The most confident new labels are added to the pool of labeled data and, then are used to retrain the models. The process keeps iterating for an indefinite time (Section 3.2 describes this process in more details). The iterative training is coupled by constraints that restrict labellings.

3.1 Coupling Constraints Used by CBS

As already mentioned, the inspiration to CBS is taken from [10, 4], where three coupling constraints are defined:

- **Output constraints:** For two functions $f_a : X \to Y_a$ and $f_b : X \to Y_b$, if we know some constraint on values y_a and y_b for an input x, we can require f_a and f_b to satisfy this constraint. For example, if f_a and f_b are Boolean-valued functions and $f_a(x) \Rightarrow f_b(x)$, we could constrain $f_b(x)$ to have value 1 whenever $f_a(x) = 1$.
- **Compositional constraints:** For two functions $f_1 : X_1 \to Y_1$ and $f_2 : X_1 \times X_2 \to Y_2$, we may have a constraint on valid y_1 and y_2 pairs for a given x_1 and any x_2. We can require f_1 and f_2 to satisfy this constraint. For example, f_1 could "type check" valid first arguments of f_2, so that $\forall x_1, \forall x_2, f_2(x_1, x_2) \Rightarrow f_1(x_1)$.
- **Multi-view-agreement constraints:** For a function $f : X \to Y$, if X can be partitioned into two "views" where we write $X = \langle X_1, X_2 \rangle$ and we assume that both X_1 and X_2 can predict Y, then we can learn $f_1 : X_1 \to Y$ and $f_2 : X_2 \to Y$ and constrain them to agree. For example, Y could be a set of possible categories for a web page, X_1 could represent the words in a page, and X_2 could represent words in hyperlinks pointing to that page (this example was used for the Co-Training setting [18]).

Considering the CBS approach, the learned functions can be considered *classifiers* informing the system whether a given noun phrase is an instance of some category (or whether a pair of noun phrases is an instance of some semantic relation).

In the experiments described in Section 4, the (*output constraint*) coupling is used to implement the *mutual exclusiveness* constraint described in Subsection 3.2. In this sense, consider that both *city* and *company* are categories. In addition, consider that *city* has been defined as mutually exclusive with *company.* In such a scenario, CBS will then have binary functions (classifiers) $f_a : X_{NE} \rightarrow Y_{city}$ and $f_b : X_{NE} \rightarrow Y_{company}$. If, for a specific noun phrase (e.g. *Bristol*), $f_a(Bristol) = 1$ and $f_b(Bristol) = 1$, then the belief that *Bristol* is a *city* (and also a *company*) decreases. However, if $f_a(Bristol) = 1$ and $f_b(Bristol)$ $= 0$, then the belief that *Bristol* is a *city* (and not a *company*) increases.

3.2 Coupled Bayesian Set Algorithm

In this section, we describe our algorithm CBS to improve semi-supervised learning for information extraction based on coupling principles. CBS was designed to address the problem of learning extractors to automatically populate categories (predefined in an initial ontology) with high-confidence instances. It has as input an initial ontology (describing categories and semantic relations), a small set of labeled instances for each category and for each semantic relation and also, a large corpus of web pages.

CBS is a bootstrapping algorithm, based on Bayesian Sets (BS) [7], that leverages mutual exclusion principle using positive examples of one category as negative examples for other ones to learn high-precision instances for all categories defined in an initial ontology.

Based on BS scoring metric (see Equations (10) and (12)), consider, that in CBS, we are simultaneously learning one classifier for each category given in the initial ontology. Assume that category C has weight vector q^c (obtained using positive labeled examples for that category) and it is mutually exclusive with K categories with $q^1, q^2 \dots q^k$ as their weight vector respectively. Then, in this case, CBS score for an instance x is evaluated as:

$$\log score(x) = c + \sum_j q_j^c x_{.j} - \sum_i \sum_j q_j^i x_{.j} \tag{14}$$

where $q_j^i = 0$ for all j which are positive features (obtained using positive labeled examples) of category q_j^c for all i and also for all j which are not features of the i^{th} class, and c is calculated as follows:

$$c = \sum_j \log(\alpha_j + \beta_j) - \log(\alpha_j + \beta_j + N) + \log \tilde{\beta}_j - \log \beta_j$$

where α_j and β_j are hyper-parameters, and

$$q_j^i = \log \tilde{\alpha}^i - \log \alpha_j - \log \tilde{\beta}_j{}^i + \log \beta_j$$

Following BS ideas, if we put the entire data set D into one large matrix X, we can compute the vector s of $\log \mathit{scores}$ for all points using a single matrix vector multiplication

$$s = c + Xq^c - \sum_i Xq^i \tag{15}$$

The hyper-parameters α and β are empirically set from data,

$$\alpha = \eta * m$$

$$\beta = \eta * (1 - m)$$

where m is a mean vector of features over all instances, and η is a scaling factor ($\eta = 2$ in our experiments).

The intuition behind Equation(14) is that it predicts high score for an instance x which has more features related only to that particular category. On the other hand, it penalizes the score of instances having a higher number of features present in mutually exclusive categories. Penalization depends upon weight vector of mutually exclusive categories. Therefore, for an instance related only with features that are exclusive to the target category (and having no relation with features that are present in other mutually exclusive categories) equation(14) reduces to the same as equation(10). In the case where the instance x is related to features that are shared by mutually exclusive categories, equation(14) can also be rewritten as given below showing reduced effective category weight vector.

$$\log \mathit{score}(x) = c + \sum_j (q_j^c - \sum_i q_j^i) x_{.j} \tag{16}$$

The main motivation is to have a classifier that gives more strength to features that are *exclusive* to a single category and penalizes features which are common among various categories. This helps to integrate the constraint information in our system and extract high confidence instances from data.

If for a given instance x, categories a and b are mutually exclusive, then both feature and weigth vectors of category a and b will be used to estimate two scores for x (score_a and score_b, respectively). For example, if category a has non zero feature ids say (1,2,4,6,8,10) for classification and (1,2,3,5,8,9,10,15) for category b, out of 15 total ids. Then, $\log \mathit{score}_a(x)$ will be calculated adding the values for features (1,2,4,6,8,10) and subtracting (penalizing) the values for features (3,5,9 and 15) according to the value of weight vector q_a and q_b respectively. All the other features (7,11,12,13,14) do not contribute to $\log \mathit{score}_a(x)$. This is the reason why we have q_j^i=0 for all j which are features of the target category (i.e in case of category a, $q_j^b = 0$ for all $j = (1, 2, 4, 6, 8, 10)$ and also for all j which are not the features of i^{th} category weight vector). And in this example, $q_j^b = 0$ for all $j = (7, 11, 12, 13, 14)$. A summarized version of CBS pseudo-code is presented in Algorithm 1.

Algorithm 1 . Coupled Bayesian Sets algorithm

```
Input: An initial ontology O (defining categories, mutually exclusiveness relations
  and a small set of labeled examples to each category) and a corpus C
Output: Trusted instances for each given category
for $i = 0$ to $\infty$ do
    for each category do
        extract new instances using available labeled examples
        filter instances which are violating coupling;
        rank instances using score mentioned in Equation (14);
        label top ranked instances;
    end for
end for
```

In CBS, instances are filtered to enforce mutual exclusion. An instance x is rejected whenever $score_{ci}(x)$ for the target category ci is lower than all $score_{cj}$ for all the other categories cj (where $j \neq i$). This soft constraint is much more tolerant of the inevitable noise in web text as well as ambiguous noun phrases than a hard constraint.

CBS was specially designed to allow efficient learning of many categories simultaneously from a very large corpus of sentences extracted from web text. Considering we have a binary sparse corpus (texts from the web), scoring all items in a large data set can be accomplished using a simple sparse matrix-vector multiplication (as done in BS). Thus, we get a very fast and simple algorithm.

4 Experiments and Results

We ran our experiments using a subset obtained from ClueWeb [19]. Our dataset consists of 2,070,896 noun phrases (*np*s) as instances and 72,996 contexts (*cont*s) as features. The dataset is stored as a $cont \times np$ matrix (context by noun phrase matrix) $\mathcal{M}$, where each cell $\mathcal{M}_{i,j}$ represents the number of co-occurrences of $cont_i$ and np_j. To transform $\mathcal{M}$ in a binary matrix, the data was preprocessed normalizing each cell value based on the sum of each column, and then thresholding so that $\mathcal{M}_{i,j} = 1$ if $(np_j - frequency) > (2 \times context - frequency - mean)$.

The input ontology used in all experiments is a subset taken from NELL's ontology and has 11 categories namely *Company*, *Disease*, *KitchenItem*, *Person*, *PhysicsTerm*, *Plant*, *Profession*, *SocioPolitics*, *Website*, *Vegetable*, *Sport*. Categories were initialized with 6-8 seed instances specified by a human.

The performed experiments were designed in order to help us having empirical evidence to answer the following question:

1. Can CBS outperform other algorithms, such as BS [7], Iterative Bayesian Sets BaS-all [9] and Coupled Pattern Learner CPL [4], in the task of category instances extraction?

2. Can CBS be applied to a task of populating NELL's ontology in an iterative bootstrap approach?
3. Can CBS be applied to a task of populating an ontology in which mutually exclusiveness relations are not known (and in such a situation, these relation can be automatically discovered and used as new constraints for coupling)?

4.1 Coupled Bayesian Sets *versus* Other Approaches

In order to have empirical evidence to answer questions (1), Coupled Bayesian set algorithm (CBS) was used to extract (from a specific corpus) category instances following the methodology described in [4]. In CBS, for each extracted instance a score is calculated (based on $equation(15)$) and then a filter is applied coupling the results of all the classifiers. Here, the mutually exclusive principle is used for coupling (i.e. an instance x can not belong to more than one mutually exclusive category). After filtering out the instances (using coupling), we promote the top 5 new instances as new labeled examples for that category. To allow comparative analysis, the same methodology was applied using the original Bayesian Sets algorithm (BS), the CPL algorithm (the category extractor used in NELL), and also the Bas-all algorithm [9]. Following along these lines, we performed 10 iterations. Top 20 output instances for sports category are shown in Table 4.2 (where incorrect output instances are highlighted).

To have a better idea of the precision of each one of the methods used in the performed experiments, a metric commonly used in set expansion evaluation [9] was adopted. This metric is referred to as $Precision@N$ and is calculated in the following way: after ranking all the promoted instances in an specific iteration, the percentage of correct instances in the subset formed by the top N entities (in the ranked list) is calculated. Table 2 shows the results after one, three, five and ten iterations. The net effect is substantial, as is apparent

Analyzing Table 2, it is possible to notice that CBS is the only method (in these experiments) that could keep precision rates above 85% even after 10 iterations. This can be considered empirical evidence that CBS can avoid concept drift in situations where other approaches would fail. BS and Bas-all achieved very low precision rates after ten iterations (below 40%). And CPL could keep good precision up to seven iteration, but its results started deteriorating after ten iterations. It is important to mention, however, that CPL is not the only algorithm employed by NELL. Thus, the results shown for CPL (in Table 2) do not represent NELL's precision. On the other hand, these result can give some evidence that CBS could help NELL's self-supervised approach to prevent concept drift. Table 3 presents the precision of different categories for CBS, BS, Bas-all and CPL.

Considering that Bayesian Sets are defined on some adaptation from the Naive Bayes classifier [20], to finish this subsection we present (see Table 4) some results from experiments performed to compare CBS and Naive Bayes algorithm using a small version of our dataset consisting of 5548 contexts and 12,500 noun phrases.

Table 1. Top 20 instances for Category Sport in the first and second iterations of CBS, BS and Bas-all

Iteration 1			**Iteration 2**		
CBS	**BS**	**BaS-all**	**CBS**	**BS**	**BaS-all**
Football	Football	football	football	golf	sports
Baseball	Baseball	baseball	Baseball	football	boxing
Basketball	basketball	Basketball	Basketball	baseball	dance
Soccer	Soccer	Soccer	Soccer	soccer	**politics**
Skiing	Skiing	Skiing	Skiing	surfing	fishing
Tennis	Tennis	Tennis	Tennis	skiing	golf
Hockey	Hockey	Hockey	Hockey	cricket	football
Swimming	swimming	Swimming	Swimming	Tennis	baseball
Wrestling	Wrestling	Wrestling	Wrestling	hockey	basketball
Boxing	Boxing	Boxing	Boxing	swimming	soccer
Volleyball	Golf	sport	Volleyball	chess	skiing
Polo	Volleyball	golf	Softball	wrestling	tennis
Badminton	Chess	fishing	Polo	boxing	hockey
Curling	Cricket	chess	Badminton	dancing	chess
table tennis	**Yoga**	cricket	table tennis	**Meditation**	swimming
water polo	surfing	**guitar**	Curling	**cooking**	wrestling
Bocce	**guitar**	dancing	cycling	**piano**	**photography**
Softball	Dancing	hunting	scuba diving	**guitar**	**yoga**
cycling	sailing	sailing	water polo	sailing	**writing**

Table 2. Precision@30 of CBS, BS, CPL and Bas-all after one, three, five and ten iterations

	Precision@30 after Iteration				
Algorithms	1^{st}	3^{rd}	5^{th}	7^{th}	10^{th}
CBS	79%	84%	92%	90%	87%
BS	68	70%	72%	54%	36%
CPL	74%	78%	79%	82%	70%
Bas-all	70%	72%	74%	64%	39%

Top 10 output instances for the very first iteration of categories countries, sports, food are shown in Table 4.

We believe that most of our results are self-explanatory, there are a few details that we would like to elaborate on. We found out that though Naive Bayes classifier can predict correctly the classes for large number of instances but the probability with which it classifies is not good enough for methods like iterative bootstrapping learning. It is evident from the Table 4 that CBS completely outperforms the Naive Bayes algorithm in our case.

Table 3. Precision@30 for CBS, BS, CPL and Bas-all in all 11 categories (after 5 and 10 iterations)

	Iteration 5				**Iteration 10**			
Categories	**CBS**	**BS**	**CPL**	**Bas-all**	**CBS**	**BS**	**CPL**	**Bas-all**
Companies	100%	78%	64%	78%	100%	44%	54%	44%
Diseases	100%	84%	100%	84%	100%	48%	74%	54%
KitchenItems	94%	92%	97%	92%	94%	40%	94%	40%
Persons	100%	64%	82%	64%	100%	32%	68%	32%
PhysicsTerms	100%	78%	82%	84%	100%	36%	78%	48%
Plants	100%	68%	94%	74%	100%	38%	84%	32%
Professions	100%	84%	84%	84%	87%	54%	87%	54%
SocioPolitics	48%	30%	38%	30%	34%	18%	28%	14%
Sports	97%	84%	90%	84%	100%	43%	87%	54%
Websites	94%	64%	67%	74%	90%	36%	58%	36%
Vegetables	83%	72%	78%	64%	48%	14%	54%	14%
Average Precision@30	**92%**	**72%**	**79%**	**74%**	**87%**	**36%**	**70%**	**39%**

Table 4. Top 10 output instances for CBS and Naive Bayes after 1^{st} iteration. Wrong extractions are highlighted.

Query:Countries		Query:Sports		Query:Food	
NB	CBS	NB	CBS	NB	CBS
Order	United States	**Fishing**	football	information	tomatoes
Argentina	China	**Guitar**	basketball	**Glass**	spinach
India	Canada	**Development**	Baseball	**advice**	fontina
Future	England	**Politics**	Soccer	**interview**	Shrimps
US	Japan	Baseball	Tennis	**manager**	Pancetta
U.S.	India	**character**	Wrestling	bread	Strawberry
cash	France	**competition**	Hockey	butter	parmesan-cheese
France	Russia	**creation**	Boxing	fruits	Coffees
South Africa	Mexico	**poker**	Softball	**list**	bread
government	Singapore	football	NFL football	**chance**	GreenOnions

4.2 CBS *versus* CPL: Beyond Concept Drift

The results presented in the previous subsection (Subsection 4.1) give empirical evidence that CBS can prevent concept drift in scenarios where other algorithms

Table 5. CPL probability and CBS score for extracted instances (after 5 iterations) for category Website

CPL	probability	CBS	score
Wikepedia	0.9375	google	1104.486
Google	0.9375	wikipedia	1104.486
radio	0.9375	facebook	877.4
page	0.9375	mysapce	806.103
yahoo	0.9375	youtube	718.338
blog	0.9375	twitter	482.433
facebook	0.9375	yahoo	457.542
monday	0.9375	Wordpress	443.554
ebay	0.875	amazon	416.628
MSN	0.875	skype	394.378

might fail. Therefore, CBS tends to be a good algorithm to perform category instances extractions in a system like NELL.

Another interesting issue related to CBS (that can make it suitable to be used in a never-ending learning system like NELL) is its capability of discriminating the probabilistic score of each extracted instance. In other words, when running a classifier based on many features (hundreds of features or more) it is common that the probabilistic score of most predictions are close to each other (tending to 0 or 1). Such a behavior is very common in the Naive Bayes classifier and also in logistic regression approaches having too many features. Considering that CBS (as well as BS) is based on the idea of marginalizing out the (unknown) parameters of the model (for each query), the algorithm tends to give a more discriminative probabilistic score for each performed prediction.

To have some empirical evidence on how CBS would perform regarding the probabilistic score precision and discriminations (when compared to CPL) some experiments were designed. Thus, a smaller dataset consisting of 5200 contexts and 68,919 instances was used as input to both CBS and CPL and results (after 5^{th} iteration) for categories Websites (see Table 4.2) and Sports (Table 4.2) extractions as well as their respective scores were analyzed.

Results are self explanatory but one important thing which we would like to point is that, for CPL results, most of instances have same probability, while in CBS results, the scores tend to discriminate each extracted instance. In order to illustrate an interesting scenario related to it, consider that, after every iteration only the top five extractions (the ones with higher confidence associated) should be promoted. In such a situation, CPL results (presented in Table 4.2) would introduce some uncertainty on which would be the best instances to be promoted (because there are 8 instances with the same probability). Results obtained using CBS, on the contrary, would not bring this uncertainty to the promotion task.

These results reflect the potential of CBS to learn through bootstrapping approach and stand robustly against BS, Naive Bayes, Bas-all and Nell.

Table 6. CPL probability and CBS score for extracted instances (after 5 iterations) for category Sport

CPL	probability	CBS	score
Game	0.998047	Baseball	1782.201
Show	0.998047	Basketball	1630.333
Football	0.998047	Soccer	1223.195
Day	0.998047	Skiing	1162.535
Drama	0.996094	Tennis	1022.093
Music	0.996094	Hockey	1012.905
Basketball	0.996094	Sailing	984.733
chess	0.992188	Wrestling	802.307
Baseball	0.992188	Boxing	724.129
Golf	0.992188	Swimming	677.489

4.3 Automatically Finding Negative Examples for Coupling

To find some insight and empirical evidence to help us answering the third question, some extra experiments were designed as follows. In this experiments, at first, we run CBS without any negative seed example ($Classifier_1$). After getting the top instances for each category (to be promoted), we also extract the bottom instances (from $Classifier_1$) as negative seed examples. Then, for each category, we create a new CBS version ($Classifier_2$), whose seed instances are these negative seed examples. Therefore, we can apply the two CBS versions ($Classifier_1$ and $Classifier_2$) as if they were classifiers for two mutually exclusive categories. Thus with this approach, we have built a new constraint relation for a category which is independent of previously known mutually exclusive relationships. For support of our above discussion we have compared this approach with BaS-all[9] which consider only positive seed examples for entity set

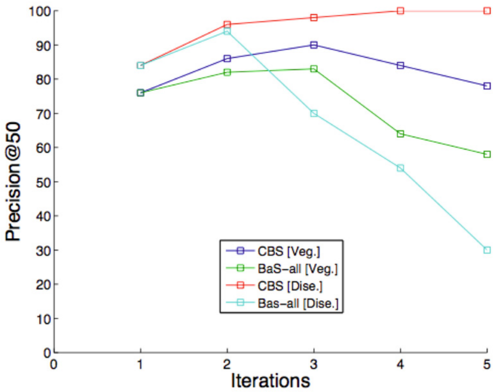


Fig. 1. Precision@50 of CBS and BaS-all algorithms over categories Vegetables and Diseases

expansion. We run CBS for two categories namely Vegetables and Diseases against BaS-all and the results are shown in Figure 1.

Analyzing Figure 1 it is possible to notice that using CBS results to generate new negative examples, which can be coupled to the extractions process, can help the system to reduce the impact of concept drift even when no mutually exclusive relation is given in advance.

5 Conclusion and Future Work

In this paper, we consider the problem of semi-supervised learning approach to extract category instances (e.g. country(USA), city(New York) from web pages, starting with a handful of labeled training examples of each category, plus hundreds of millions of unlabeled web documents (as described in NELL [10]. Following along these lines, we propose a new algorithm, based on Bayesian Sets [7], to perform a set expansion task which can help a never-ending learning system (such as NELL) to avoid concept drifting during the iterative (and never-ending) process of extracting facts from the Web. The proposed algorithm is named Coupled Bayesian Sets (CBS). CBS implementation makes it fast as its only need to perform a sparse Matrix×Vector multiplication, and thus, it can easily be applied to huge data collections. The performed experiments revealed that CBS can outperform algorithms such as the original Bayesian Set, the Naive Bayes classifier, the Bas-all and the coupled semi-supervised logistic regression algorithm (CPL) on which Nell is currently running. In addition, CBS can be used to automatically generate new constraints to the set expansion task even when no mutually exclusiveness relationship is previously defined, thus, allowing the method to help NELL's self-reflection capabilities. As future work we intend to adjust and evaluate CBS for exploring also other types of coupling constraints such as Compositional and Multi-view-agreement constraints. We would also like to use CBS in the Portuguese version of Nell which is currently under development.

Acknowledgements. This work was partially supported by Department of Science and Technology, Government of India under Indo-Brazil cooperation programme and the Brazilian research agencies CAPES, CNPq and FAPESP. Also we are very grateful to Zoubin Ghahramani and K.A. Heller for making available their Bayesian set code for comparison. In addition, we would like to thank Tom Mitchell and all the Read The Web group for helping on issues related to NELL. We also place our thanks on record to all those who have directly or indirectly helped us in completion of this paper.

References

1. Bikel, D.M., Schwartz, R., Weischedel, R.M.: An algorithm that learns what's in a name. Machine Learning 34(1), 211–231 (1999)
2. Talukdar, P.P., Pereira, F.: Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In: ACL 2010, pp. 1473–1481 (2010)

3. Pennacchiotti, M., Pantel, P.: Automatically building training examples for entity extraction. In: Proceedings of Computational Natural Language Learning (CONLL 2011), pp. 163–171 (2011)
4. Carlson, A., Betteridge, J., Wang, R.C., Hruschka Jr., E.R., Mitchell, T.M.: Coupled semi-supervised learning for information extraction. In: Proc. of WSDM (2010)
5. Riloff, E., Jones, R.: Learning dictionaries for information extraction by multi-level bootstrapping. In: Proc. of AAAI (1999)
6. Curran, J.R., Murphy, T., Scholz, B.: Minimising semantic drift with mutual exclusion bootstrapping. In: Proc. of PACLING (2007)
7. Ghahramani, Z., Heller, K.: Bayesian sets. In: Advances in Neural Information Processing Systems, vol. 18 (2005)
8. Sadamitsu, K., Saito, K., Imamura, K., Kikui, G.: Entity set expansion using topic information. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers, HLT 2011, vol. 2, pp. 726–731. Association for Computational Linguistics, Stroudsburg (2011)
9. Zhang, L., Liu, B.: Entity set expansion in opinion documents. In: Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia, HT 2011, pp. 281–290. ACM, New York (2011)
10. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Proceedings of the Twenty-Fourth Conference on Artificial Intelligence, AAAI 2010 (2010)
11. Brin, S.: Extracting patterns and relations from the world wide web. In: Proc. of WebDB Workshop at 6th Int. Conf. on Extending Database Technology (1998)
12. Collins, M., Singer, Y.: Unsupervised models for named entity classification. In: Proc. of EMNLP (1999)
13. Agichtein, E., Gravano, L.: Snowball: extracting relations from large plain-text collections. In: ACM DL, pp. 85–94 (2000)
14. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised named-entity extraction from the web: an experimental study. Artif. Intell. 165(1), 91–134 (2005)
15. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: IJCAI (2007)
16. Etzioni, O., Fader, A., Christensen, J., Soderland, S., Mausam: Open information extraction: The second generation. In: IJCAI, pp. 3–10 (2011)
17. Hoffart, J., Suchanek, F.M., Berberich, K., Lewis-Kelham, E., de Melo, G., Weikum, G.: Yago2: exploring and querying world knowledge in time, space, context, and many languages. In: Proc. of the 20th Int. Con. on World Wide Web, WWW 2011, pp. 229–232. ACM, New York (2011)
18. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proc. of COLT (1998)
19. Callan, J., Hoy, M.: Clueweb09 data set (2009), http://boston.lti.cs.cmu.edu/Data/clueweb09/
20. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. John Wiley & Sons Inc. (June 1973)

Graph-Based Transduction with Confidence

Matan Orbach and Koby Crammer

Dept. of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel
{matanorb@tx,koby@ee}.technion.ac.il

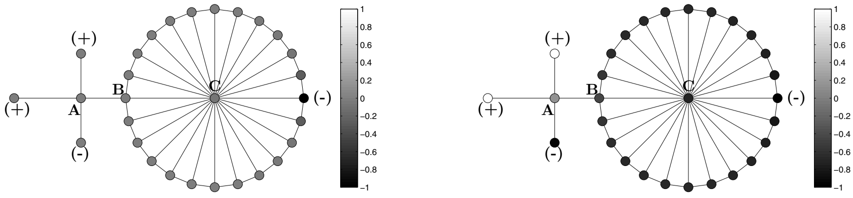
Abstract. We present a new multi-class graph-based transduction algorithm. Examples are associated with vertices in an undirected weighted graph and edge weights correspond to a similarity measure between examples. Typical algorithms in such a setting perform label propagation between neighbours, ignoring the quality, or estimated quality, in the labeling of various nodes. We introduce an additional quantity of confidence in label assignments, and learn them jointly with the weights, while using them to dynamically tune the influence of each vertex on its neighbours. We cast learning as a convex optimization problem, and derive an efficient iterative algorithm for solving it. Empirical evaluations on seven NLP data sets demonstrate our algorithm improves over other state-of-the-art graph-based transduction algorithms.

1 Introduction

Supervised machine learning algorithms are powerful. Given a training set composed of example-label pairs, many methods have been proposed and successfully used for a variety of real-world tasks in different domains. Typically, performance improves as the size of the training set increases. However, this improvement comes with a price, labeling a large amount of data is slow, costly and prone to human errors, especially in complex tasks. This is in contrast to the fact that a large amount of unlabeled data can be cheaply collected in many different domains. As a result, it has become beneficial to research and develop semi-supervised learning (SSL) algorithms. These algorithms are designed to learn from a small set of labeled data and a much larger set of unlabeled data.

Graph-based methods are an important class of SSL algorithms [3,11,12,14]. These methods construct an undirected weighted graph reflecting a similarity relation between examples, both labeled and unlabeled. Each example is associated with a vertex. Edge weights correspond to a similarity measure between examples. Learning is then based on a *smoothness assumption* [3]: two neighbour vertices are likely to have the same label. In this paper we consider the transductive graph-based setting. The goal of the learning algorithm is to assign labels to the unlabeled vertices of the graph. Starting with the small set of labeled vertices, the learning algorithm propagates label information from the small set of known labels to the rest of the graph. In other settings, algorithms learn (also) a model and are thus also capable of classifying new, previously unseen, examples.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 323–338, 2012.



(a) MAD

(b) Ours (TACO)

Fig. 1. Illustration of label-propogation properties for two algorithms : MAD [12] and ours (named TACO). MAD is sensitive to vertex-degree differences while our algorithm is robust to it.

Two new transductive graph-based algorithms have been introduced recently: Adsorption [1] and Modified Adsorption (MAD) [12]. The label propagation in Adsorption can be viewed as a controlled random walk over the graph. The transition probabilities in the random walk are determined by edge weights and the degree of each vertex. Specifically, transition probabilities to and from high degree vertices are reduced, assuming a larger neighbours set is more likely to contain disagreements among the neighbours. We further discuss this assumption shortly. MAD builds upon Adsorption, formulating a convex optimization problem including the controlled random walk transition probabilities, and solving this problem with an efficient iterative algorithm.

We present a new multi-class graph-based transductive algorithm. The main motivation for our new algorithm is the following question: should we always discourage high degree vertices? While both Adsorption and MAD are based on a definite positive answer (and label propagation (LP) [14] is somewhat based on an implicit negative answer), our goal is aimed at letting the data decide for us. Assume we have a high degree vertex, and our label propagation algorithm is in a state where almost all of its neighbours have the same estimated label. Here, we can be highly confident in our estimated label because we have many agreeing neighbours. Thus, in this case, the influence of the high degree vertex on its neighbours should not be reduced. Only when we have a high degree vertex, combined together with a large measure of disagreement among neighbours, we should lower the effect of this vertex on its neighbours.

Our motivation is illustrated in Fig. 1, showing a simple graph for a binary label propagation problem. Labeled vertices with positive class are marked with (+) (white) and negative class with (−) (black). Vertex **A** has low degree and a large measure of disagreement among its neighbours, and vertex **C** has high degree and is connecting many other vertices (in the middle of a cluster). Vertex **B** is member of the cluster, and is also connected to vertex **A**. Gray level indicates the assigned label value by the algorithms. Our intuition implies that the correct action will be to lower the effect of the confusing **A**, while propagating the negative label from right to left through **C** to **B** and the entire cluster. MAD lowers the effect of **C**, thus damaging the ability to identify all vertices in the cluster. In contrast, our algorithm (named TACO below) correctly identifies the cluster while ignoring the unreliable vertex **A**.

To measure the level of agreement between neighbours, as well as account for vertex degree, our algorithm maintains per vertex confidence information in addition to label information. The confidence parameters capture the quality of the estimated label information. One consequence of that model is that the algorithm does not reduce the effect of *all* high degree vertices, but only the effect of vertices with low confidence in their estimated labels. Both confidence and label information are propagated throughout the graph.

After introducing our model we cast learning as a convex optimization problem and propose an efficient algorithm for solving it. We experiment with seven text categorization problems of various sizes and show that our algorithm, evaluated with several metrics, outperforms other algorithms (MAD and alternating minimization (AM) [10]) on most tasks.

2 Problem Formulation

We focus on transductive learning using graphs. The input is constituted of two sets: a set of n_l labeled examples $D_l = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_l}$ and a set of n_u unlabeled examples $D_u = \{\boldsymbol{x}_i\}_{i=n_l+1}^{n_l+n_u}$. Examples belong to some input space $\boldsymbol{x}_i \in \mathcal{X}$, where we assume in this paper that it is a vector space $\mathcal{X} = \mathbb{R}^d$. Labels belong to a finite label set $y_i \in L$ denoted by $L = \{1, \ldots, m\}$. The goal of the learning algorithm is to assign a label $\hat{y}_i \in L$ to each of the (unlabeled) examples in D_u. We denote the total number of examples by $n = n_l + n_u$.

In order to be able to propagate the labels from the labeled examples to the unlabeled ones, we assume the existence of an undirected weighted graph $G = (V, E, W)$ over all input examples. Each input example $\boldsymbol{x}_i$ is associated with a vertex $v_i \in V$. For ease of presentation we assume that the labeled input examples are associated with the first n_l vertices in V and we refer to them as *labeled vertices*. Similarly, the remaining n_u vertices in V are associated with the unlabeled input examples, and are called *unlabeled vertices*.

A weighted edge $e \in E = V \times V$ represents label-similarity. The larger the weight $w_{i,j} \in W$ of an edge between vertices v_i and v_j is, the more we believe the labels of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ should be the same. We assume that the weight matrix $W \in \mathbb{R}^{n \times n}$ is symmetric with non-negative elements, and that there is an edge between any two vertices, although in practice most edges will have the lowest weight of zero.

We denote by $\delta_l(i) = \mathbf{1}_{[i \leq n_l]}$ the indicator of a vertex to be labeled, that is $\delta_l(i) = 1$ iff the vertex v_i is a labeled vertex. We associate a labels vector $\boldsymbol{y}_i \in \{0, 1\}^m$ with each vertex. For vertices associated with labeled examples v_i we set $y_{i,r} = 1$ iff the correct label of example $\boldsymbol{x}_i$ is $y_i = r$. All other entries are set to 0. For vertices associated with unlabeled examples v_j we set $\boldsymbol{y}_j = \mathbf{0}$, the vector with all elements equal to zero. We denote the complete prior information matrix by $\mathbf{Y} \in \mathbb{R}^{n \times m}$, where the i*th* row $\boldsymbol{y}_i$ corresponds to vertex v_i.

3 Algorithm

We cast learning as an optimization over two sets of parameters, each contains one element per vertex v_i in the graph. The first set of parameters the algorithm maintains is a per vertex score vector $\boldsymbol{\mu}_i = [\mu_{i,1}, \ldots, \mu_{i,m}]^\top \in \mathbb{R}^m$. The larger the r$th$ element $\mu_{i,r}$ is, the stronger is our belief that the input $\boldsymbol{x}_i$ associated with vertex v_i belongs to class r. The final predicted label is given according to the highest score, namely $\hat{y}_i = \arg\max_r \mu_{i,r}$.

In addition, the algorithm maintains a diagonal non-negative matrix per vertex, $\boldsymbol{\Sigma}_i \in \mathbb{R}^{m\times m}$, where we denote by $\sigma_{i,r}$ the rth diagonal element of $\boldsymbol{\Sigma}_i$. Each parameter $\sigma_{i,r}$ is associated with our uncertainty in the corresponding score parameter $\mu_{i,r}$. The lower the value of $\sigma_{i,r}$ is, the higher our confidence in the score value $\mu_{i,r}$.

Conceptually, the score vectors are our *first order* information over labels, while the uncertainty matrices are our *second order* information. Most, if not all, previous graph-based transduction algorithms maintain only first order information (scores) over vertices, ignoring agreement or disagreement between neighbour vertices. The second order information is designed to capture this exact information, allowing the algorithm to better label various vertices by considering agreement levels among their neighbours.

Next, we formulate an unconstrained convex optimization problem in the variables $\{(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_{i=1}^n$ as our learning objective, and derive an efficient iterative algorithm for minimizing it.

3.1 Objective

We have three desired properties of the optimization problem used to define learning, the first two properties build on previous research [12,14], while the last is required for the usage of confidence.

First, a pair of close vertices v_i and v_j (i.e. with large $w_{i,j}$) should have close score vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ if we are certain in both vectors, that is if both matrices $\boldsymbol{\Sigma}_i$ and $\boldsymbol{\Sigma}_j$ have low eigen-values. If at least one of these matrices has large eigen-values, then we are not confident in the score values of at least one of the corresponding vertices, and therefore relax the demand that the two score vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ should be close.

Second, the score vectors of labeled vertices should be close to the true label vectors associated with them, again, if the corresponding uncertainty is low (or large certainty). In other words, if the eigenvalues of $\boldsymbol{\Sigma}_i$ are low, then $\boldsymbol{\mu}_i \approx \boldsymbol{y}_i$.

Third, the uncertainty should be far from infinity and not close to zero. As in both cases, the first two properties would be invalid. Specifically, we add a term that drives the uncertainty close to some predefined values, one can think of this term as a combination of both regularization (far from infinity) and barrier (strictly greater than zero) for uncertainty.

We formalize the above intuition using a symmetric Mahalanobis distance that is based on the uncertainty matrices. Specifically, given two pairs $(\boldsymbol{x}, \mathbf{S})$ and $(\boldsymbol{y}, \mathbf{T})$ of score vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$ and uncertainty matrices $\mathbf{S}, \mathbf{T} \in \mathbb{R}^{m \times m}$ we define the squared-distance between them to be,

$$D_M\left(\boldsymbol{x}, \boldsymbol{y}, \mathbf{S}, \mathbf{T}\right) = \left(\boldsymbol{x} - \boldsymbol{y}\right)^\top \left(\mathbf{S}^{-1} + \mathbf{T}^{-1}\right)\left(\boldsymbol{x} - \boldsymbol{y}\right) \ . \tag{1}$$

If either $\mathbf{S}$ or $\mathbf{T}$ have large eigenvalues, then the distance is low even if $\boldsymbol{x}$ and $\boldsymbol{y}$ are not close to each other. On the other hand, if both $\mathbf{S}$ and $\mathbf{T}$ have low eigenvalues, then in order for the distance to be low, we require that $\boldsymbol{x}$ and $\boldsymbol{y}$ would be close to each other.

Common [11,12,14] choice for formulating the first property is to require that neighbour vertices would have close scores, that is,

$$\sum_{i,j=1}^{n} w_{i,j} D\left(v_i, v_j\right) \tag{2}$$

where $D\left(v_i, v_j\right)$ is some distance function measuring the difference between scores for v_i and v_j.

The second desired property is that the scores for labeled vertices should be close to the input labels of these vertices. We formulate this requirement using the same conceptual ideas of the first property. We view each labeled vertex as two vertices, denoted by v_i and z_i. As before, the vertex v_i is associated with scores $\boldsymbol{\mu}_i$ and uncertainty $\boldsymbol{\Sigma}_i$. The new vertex z_i has *fixed* scores $\boldsymbol{y}_i$ and *fixed* uncertainty $\frac{1}{\gamma}\mathbf{I} \in \mathbb{R}^{m \times m}$ for some $\gamma > 0$. The new vertex z_i with fixed parameters is connected only to its counterpart v_i with an edge of weight 1. Similarly to (2) we add a second term to our objective capturing the second property,

$$\sum_{i=1}^{n_l} D\left(v_i, z_i\right) \ . \tag{3}$$

The third and last property is formally a regularization term forcing the uncertainty matrix to be close to some predefined matrix. We use the following convex function which is a sum of two terms, the first monotonic in the eigenvalues, and the second prevents matrices from having zero eigenvalues,

$$\sum_{i=1}^{n} \mathrm{Tr}\boldsymbol{\Sigma}_i - \eta \sum_{i=1}^{n} \log\det\boldsymbol{\Sigma}_i \ , \tag{4}$$

for some $\eta > 0$. Clearly, the minimizers of the last terms are the matrices $\boldsymbol{\Sigma}_i = \eta\mathbf{I}$.

Combining (2) (3) and (4) together with weights $\nu, \kappa, \alpha \geq 0$ and using our distance (1) we get the final objective,

$$\begin{aligned}C\left(G,\{\boldsymbol{\mu}_i\},\{\boldsymbol{\Sigma}_i\}\right) =&\frac{1}{4}\nu\sum_{i,j=1}^{n} w_{i,j}\left[\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right)^\top\left(\boldsymbol{\Sigma}_i^{-1}+\boldsymbol{\Sigma}_j^{-1}\right)\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right)\right]\\ &+\frac{1}{2}\kappa\sum_{i=1}^{n_l}\left[\left(\boldsymbol{\mu}_i-\boldsymbol{y}_i\right)^\top\left(\boldsymbol{\Sigma}_i^{-1}+\frac{1}{\gamma}\mathbf{I}\right)\left(\boldsymbol{\mu}_i-\boldsymbol{y}_i\right)\right]\\ &+\alpha\sum_{i=1}^{n}\mathrm{Tr}\boldsymbol{\Sigma}_i-\alpha\eta\sum_{i=1}^{n}\log\det\boldsymbol{\Sigma}_i\ .\end{aligned}$$

We can divide the last equation by one of the constants (e.g. ν), set in practice the other constant to be $\kappa = 1$ and denote by $\beta = \alpha\eta$, ending up with the following,

$$\begin{aligned}C\left(G,\{\boldsymbol{\mu}_i\},\{\boldsymbol{\Sigma}_i\}\right) =&\frac{1}{4}\sum_{i,j=1}^{n} w_{i,j}\left[\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right)^\top\left(\boldsymbol{\Sigma}_i^{-1}+\boldsymbol{\Sigma}_j^{-1}\right)\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right)\right]\\ &+\frac{1}{2}\sum_{i=1}^{n_l}\left[\left(\boldsymbol{\mu}_i-\boldsymbol{y}_i\right)^\top\left(\boldsymbol{\Sigma}_i^{-1}+\frac{1}{\gamma}\mathbf{I}\right)\left(\boldsymbol{\mu}_i-\boldsymbol{y}_i\right)\right]\\ &+\alpha\sum_{i=1}^{n}\mathrm{Tr}\boldsymbol{\Sigma}_i-\beta\sum_{i=1}^{n}\log\det\boldsymbol{\Sigma}_i\ .\end{aligned}\tag{5}$$

We conclude this section by noting that the above objective is convex in all arguments. Thus, any algorithm for solving convex problems, like gradient-descent, can be used. In the next section we propose an efficient iterative algorithm for specifically solving (5). We then note the connections between the derived algorithm and our initial intuition.

3.2 An Iterative Algorithm

We now present an efficient algorithm for minimizing (5). The algorithm is iterative in nature. Let $\boldsymbol{\mu}_i^{(t)}$ and $\boldsymbol{\Sigma}_i^{(t)}$ denote the score vector and confidence matrix maintained by our algorithm at iteration t for vertex v_i. Roughly speaking, on each iteration t, the algorithm optimizes the objective over scores $\boldsymbol{\mu}_i$ given the score values of all other vertices $\boldsymbol{\mu}_j^{(t-1)}$ for $j \neq i$ and all uncertainty matrices $\boldsymbol{\Sigma}_j^{(t-1)}$ (for all j), and optimizes all the uncertainty matrices $\{\boldsymbol{\Sigma}_i\}$ given all the scores $\{\boldsymbol{\mu}_i^{(t-1)}\}$.

We first develop the update step for $\boldsymbol{\mu}_i^{(t)}$ and then follow with the update step for $\boldsymbol{\Sigma}_i^{(t)}$. Setting to zero the derivative of (5) with respect to $\boldsymbol{\mu}_i$ we get,

$$\begin{aligned}&\sum_{\substack{j\neq i\\ j=1}}^{n} w_{i,j}\left[\left(\boldsymbol{\Sigma}_i^{(t-1)}\right)^{-1}+\left(\boldsymbol{\Sigma}_j^{(t-1)}\right)^{-1}\right]\left(\boldsymbol{\mu}_i^{(t)}-\boldsymbol{\mu}_j^{(t-1)}\right)\\ &+\delta_l(i)\left[\left(\boldsymbol{\Sigma}_i^{(t-1)}\right)^{-1}+\frac{1}{\gamma}\mathbf{I}\right]\left(\boldsymbol{\mu}_i^{(t)}-\boldsymbol{y}_i\right)=0\ .\end{aligned}\tag{6}$$

For simplicity we introduce the following notation,

$$\mathbf{K}_{i,j}^{(t-1)} = w_{i,j}\left[\left(\boldsymbol{\Sigma}_i^{(t-1)}\right)^{-1} + \left(\boldsymbol{\Sigma}_j^{(t-1)}\right)^{-1}\right], \quad \mathbf{P}_i^{(t-1)} = \delta_l(i)\left[\left(\boldsymbol{\Sigma}_i^{(t-1)}\right)^{-1} + \frac{1}{\gamma}\mathbf{I}\right] \ . \tag{7}$$

Substituting (7) in (6) we get the following equation relating the optimal solution $\boldsymbol{\mu}_i^{(t)}$ to all other quantities, $\left(\sum_{\substack{j\neq i\\ j=1}}^{n} \mathbf{K}_{i,j}^{(t-1)}\right)\boldsymbol{\mu}_i^{(t)} + \mathbf{P}_i^{(t-1)}\boldsymbol{\mu}_i^{(t)} = \sum_{\substack{j\neq i\\ j=1}}^{n} \mathbf{K}_{i,j}^{(t-1)}\boldsymbol{\mu}_j^{(t-1)}$ $+ \mathbf{P}_i^{(t-1)}\boldsymbol{y}_i$. Solving for $\boldsymbol{\mu}_i^{(t)}$ we obtain,

$$\boldsymbol{\mu}_i^{(t)} = \left(\sum_{\substack{j\neq i\\ j=1}}^{n} \mathbf{K}_{i,j}^{(t-1)} + \mathbf{P}_i^{(t-1)}\right)^{-1}\left(\sum_{\substack{j\neq i\\ j=1}}^{n} \mathbf{K}_{i,j}^{(t-1)}\boldsymbol{\mu}_j^{(t-1)} + \mathbf{P}_i^{(t-1)}\boldsymbol{y}_i\right) \ . \tag{8}$$

Note that $\boldsymbol{\mu}_i^{(t)}$ is a matrix weighted average of all the other score vectors $\left\{\boldsymbol{\mu}_j^{(t-1)}\right\}_{j\neq i}$, and the true labels $\boldsymbol{y}_i$ (for a labeled vertex). The matrix-weights are exactly the inverse of the uncertainty matrices.

Next, we develop the update for the confidence matrices. Setting to zero the derivative of (5) with respect to $\boldsymbol{\Sigma}_i$ we get,

$$\begin{aligned}
&-\frac{1}{2}\sum_{j=1}^{n} w_{i,j}\left[\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{\mu}_j^{(t-1)}\right)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{\mu}_j^{(t-1)}\right)^{\top}\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\right]\\
&-\frac{1}{2}\delta_l(i)\left[\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{y}_i\right)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{y}_i\right)^{\top}\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\right]\\
&+\alpha\mathbf{I} - \beta\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1} = 0 \ .
\end{aligned} \tag{9}$$

We define the following positive semi-definite matrix,

$$\begin{aligned}
\mathbf{R}_i^{(t-1)} = &\frac{1}{2}\sum_{j=1}^{n} w_{i,j}\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{\mu}_j^{(t-1)}\right)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{\mu}_j^{(t-1)}\right)^{\top}\\
&+\frac{1}{2}\delta_l(i)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{y}_i\right)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{y}_i\right)^{\top}
\end{aligned} \tag{10}$$

and rewrite (9) as $-\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\mathbf{R}_i^{(t-1)}\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1} + \alpha\mathbf{I} - \beta\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1} = 0$. Multiplying both sides by $\boldsymbol{\Sigma}_i^{(t)}$ leads to a quadratic matrix equation in $\boldsymbol{\Sigma}_i^{(t)}$:

$$\alpha\left(\boldsymbol{\Sigma}_i^{(t)}\right)^2 - \beta\boldsymbol{\Sigma}_i^{(t)} - \mathbf{R}_i^{(t-1)} = 0 \ .$$

This is matrix quadratic equation with solution (as in scalars),

$$\boldsymbol{\Sigma}_i^{(t)} = \frac{\beta}{2\alpha}\mathbf{I} + \frac{1}{2\alpha}\left(\beta^2\mathbf{I} + 4\alpha\mathbf{R}_i^{(t-1)}\right)^{\frac{1}{2}} \ . \tag{11}$$

Both updates (8) and (11) are valid for full matrices $\mathbf{\Sigma}_i$. Diagonal matrices are obtained by diagonalizing the update (11) or specifically, taking the diagonal elements of the matrix $\mathbf{R}_i^{(t-1)}$ defined in (10). Thus, denoting the diagonal elements by $\mathbf{\Sigma}_i^{(t)} = diag\left(\sigma_{i,1}^{(t)}, \ldots, \sigma_{i,m}^{(t)}\right)$ and substituting into (8) we obtain the following update step for each score $\mu_{i,r}^{(t)}$:

$$\mu_{i,r}^{(t)} = \frac{\sum_{\substack{j\neq i\\ j=1}}^{n} w_{i,j}\left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\sigma_{j,r}^{(t-1)}}\right)\mu_{j,r}^{(t-1)} + \delta_l(i)\left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\gamma}\right)y_{i,r}}{\sum_{\substack{j\neq i\\ j=1}}^{n} w_{i,j}\left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\sigma_{j,r}^{(t-1)}}\right) + \delta_l(i)\left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\gamma}\right)} \tag{12}$$

Similarly, using (11) we obtain an update step for each $\sigma_{i,r}^{(t)}$:

$$\sigma_{i,r}^{(t)} = \frac{\beta}{2\alpha} + \frac{1}{2\alpha}\sqrt{\beta^2 + 2\alpha\left[\sum_{j=1}^{n} w_{i,j}\left(\mu_{i,r}^{(t-1)} - \mu_{j,r}^{(t-1)}\right)^2 + \delta_l(i)\left(\mu_{i,r}^{(t-1)} - y_{i,r}\right)^2\right]} \tag{13}$$

Note both updates (12) and (13) are separable in the labels and involve only variables with one specific label index r. In fact, if the confidence matrices are forced to be diagonal, the objective (5) can be decomposed into m separate optimization problems in independent parameters. In the experiments below we evaluated both the diagonal and full versions of the algorithm (updates (8) and (11)) and found no advantage for full confidence matrices over diagonal ones. We thus restrict ourself to diagonal matrices.

We call our algorithm *TACO* for *Transduction Algorithm with COnfidence*, and its pseudocode is summarized in Figure 2. The algorithm iterates over vertices, updating both parameters for each vertex, until some convergence criteria is met. In practice, we ran the algorithm for not more than 10 iterations.

As mentioned above for full matrices, the update of (12) sets $\mu_{i,r}^{(t)}$ to be a weighted average of neighbouring scores for each label r (and the true label if given), where each weight is a product of the edge-weight and a correction factor that depends on estimated confidence parameters. The more certain we are in a neighbour, the higher relative weight it will have.

Looking at the update step for the uncertainty matrices (either full in (11) or diagonal (13)) we note that $\sigma_{i,r}^{(t)}$ is monotonic on a quadratic measure of disagreement between neighbours, $\sum_{j=1}^{n} w_{i,j}\left(\mu_{i,r}^{(t-1)} - \mu_{j,r}^{(t-1)}\right)^2 + \delta_l(i)\left(\mu_{i,r}^{(t-1)} - y_{i,r}\right)^2$. In addition, this measure is *not* normalized using the degree of v_i, implying that for high degree vertices this measure is more likely to have a high-value, lowering the influence of the high degree vertex on its neighbours. However, this happens only when combined together with neighbours disagreement. If the score of all neighbour nodes is about the same, even if the number of them is very large, the certainty would be large (or uncertainty low).

```
**Parameters:** $\alpha > 0$, $\beta > 0$,$\gamma > 0$
**Input:** A graph $G = (V, E, W)$ and prior labeling $\boldsymbol{y}_i$ for all $v_i \in V$
**Initialize:** $t = 1$, $\boldsymbol{\mu}_i^{(0)} = \mathbf{0}$ and $\boldsymbol{\Sigma}_i^{(0)} = \mathbf{I}$ for all $v_i \in V$
**Repeat**
  – For $v_i \in V$:
      • Compute $\boldsymbol{\mu}_i^{(t)}$ from $\boldsymbol{\mu}_j^{(t-1)}$ and $\boldsymbol{\Sigma}_j^{(t-1)}$ :
          $\mu_{i,r}^{(t)} = \frac{\sum_{\substack{j\neq i\\ j=1}}^{n} w_{i,j}\left(\frac{1}{\sigma_{i,r}^{(t-1)}}+\frac{1}{\sigma_{j,r}^{(t-1)}}\right)\mu_{j,r}^{(t-1)} + \delta_l(i)\left(\frac{1}{\sigma_{i,r}^{(t-1)}}+\frac{1}{\gamma}\right) y_{i,r}}{\sum_{\substack{j\neq i\\ j=1}}^{n} w_{i,j}\left(\frac{1}{\sigma_{i,r}^{(t-1)}}+\frac{1}{\sigma_{j,r}^{(t-1)}}\right) + \delta_l(i)\left(\frac{1}{\sigma_{i,r}^{(t-1)}}+\frac{1}{\gamma}\right)} \qquad (12)$
      • Compute $\boldsymbol{\Sigma}_i^{(t)}$ from $\boldsymbol{\mu}_j^{(t-1)}$ :
          $\sigma_{i,r}^{(t)} = \frac{\beta}{2\alpha} + \frac{1}{2\alpha}\sqrt{\beta^2 + 2\alpha\left[\sum_{j=1}^{n} w_{i,j}\left(\mu_{i,r}^{(t-1)} - \mu_{j,r}^{(t-1)}\right)^2 + \delta_l(i)\left(\mu_{i,r}^{(t-1)} - y_{i,r}\right)^2\right]} \qquad (13)$
  – $t \leftarrow t + 1$
**Until convergence**
**Output:** Score vectors $\boldsymbol{\mu}_i^{(t)}$ and confidence matrices $\boldsymbol{\Sigma}_i^{(t)}$.
```

Fig. 2. The TACO algorithm for graph-based transduction

4 Empirical Evaluation

We evaluate our algorithm along with two other state-of-the-art graph-based transduction algorithms: Alternating Minimization (AM) [11] and Modified Adsorption (MAD) [12]. Both were empirically shown to outperform LP [12,11].

4.1 Data Sets

We evaluate our algorithm using seven NLP data sets summarized in Table 1.

WebKB: World Wide Knowledge Base is a text categorization data set previously used for the evaluation of several transductive algorithms [7,8,11,12]. Documents in the data set are web pages from four academic web domains, categorized according to domain and topic. We used documents from four topic categories *course*, *faculty*, *project* and *student*, for a total of 4, 199 documents. We construct the graph by first removing all non-textual information (HTML tags), followed by lower casing all tokens, computing TFIDF features, and lastly using cosine similarity to form edge weights. This graph yields better results for all evaluated algorithms in comparison to the graph used previously [11,12] where both textual and non-textual information was used.

20 Newsgroups: The dataset contains a collection of $18,828$ newsgroup documents partitioned across 20 different newsgroups[1]. We select one-quarter of the documents from each category, yielding a total of $4,715$ documents. Graph construction is as described for WebKB except for HTML removal.

Sentiment: Sentiment classification of book reviews from Amazon. Each review is labeled with stars corresponding to the opinion of the author. The range of the stars (labels) is from one (low rating) to five (high rating). We used $5,000$ reviews. Graph construction follows TFIDF features along with cosine similarity. In addition to the aforementioned data sets we use the following data of Crammer et al [4]. This data contains preprocessed feature vectors generated for several NLP data sets. Further details concerning feature extraction per data set can be found in [4]. For each data set, we computed TFIDF features from the given counts, and constructed the graph using cosine similarity.

Table 1. A summary of data sets used in empirical evaluation

Data set	Instances	Labels	Test set size
WebKB	4,199	4	3,148
20 News	4,175	20	3,534
Sentiment	5,000	5	3,750
Reuters	4,000	4	3,000
Enron A	3,019	10	2,262
Enron B	3,171	10	2,376
Amazon3	7,000	3	5,250

Reuters: Topic classification of newswire stories (RCV1-v2) [9]. We used 4,000 instances, categorized by the following general topics: *corporate*, *economic*, *government* and *markets*. **Enron:** A collection of e-mails from over 100 different users organized in folders[2]. The task is automatic classification of e-mails into folders. E-mails from two users are used: farmer-d (Enron A) and kaminski-v (Enron B). Enron A contains 3,019 instances and Enron B has 3,171 instances. For each user there are 10 labels - email folders. **Amazon3:** Product reviews from Amazon. The task is to classify reviews to one of the 3 product domains: *books*, *dvds* and *music*. 7,000 reviews are used.

4.2 Experimental Setup

We follow previous experimental setup [11,12]. From the initial input graph we construct a K Nearest Neighbour (K-NN) graph, by keeping for every vertex only its K closest neighbours. The result of this preprocessing step is a directed graph. We then removed the direction of edges, ending up with an undirected graph, where degrees of edges may be larger than K. Next, we assign labels to randomly sampled n_l documents, under the constraint that each class is represented by at least one sample in the labeled set. Finally, we sample three-quarters of the remaining documents to constitute the test set. This process gives a single transduction set, and we repeat it 21 times.

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

[2] http://www.cs.cmu.edu/~enron/

We use the first transduction set for hyper-parameter tuning, and the others for evaluation. The hyper-parameters search grid for each of the evaluated algorithms is as follows: for AM α, μ, and ν as described in [11], for MAD μ_1, μ_2 and μ_3 as described in [12] (Sec. 6.1) and for TACO $\alpha, \beta \in \{1\text{e-}8, 1\text{e-}4, 1\text{e-}2, 1, 10, 100\}$ and $\gamma \in \{1, 2, 5\}$. We search over same range of K for all algorithms: $K \in \{100, 500, 1000, 2000\}$. We found that setting the maximum number of iterations per single run to 10, for all evaluated algorithms, was sufficient for convergence.

We perform class prior normalization (CPN) by column normalizing the prior information matrix $\mathbf{Y}$ for MAD and TACO. This ensures the total initial input score is the same for all classes and reduces the advantage common classes have on rare classes during label propagation. Since the rows of $\mathbf{Y}$ for AM are probability distribution, it is not possible to perform CPN prior to running AM, and it was not used when experimenting with AM before [11].

4.3 Evaluation Metrics

We evaluate the results of our experiments using five evaluation metrics: macro-averaged Precision-Recall Break Even Point (PRBEP), accuracy (ACC), macro-averaged accuracy (M-ACC), Mean Reciprocal Rank (MRR) and macro-averaged Mean Reciprocal Rank (M-MRR).

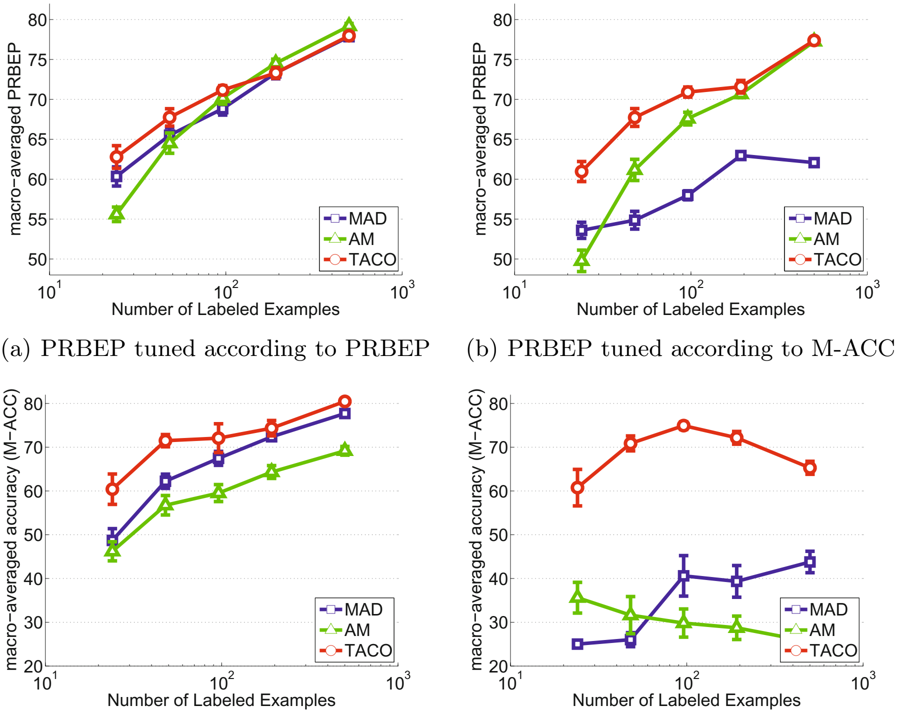
PRBEP is defined as the point in which precision and recall are equal. For each label $r \in L$ we compute PRBEP as follows. Let $\boldsymbol{s}_r = [\mu_{1,r}, \ldots, \mu_{n,r}]$ denote a vector containing the score assigned for all vertices and the label r. We define a threshold τ and use it for prediction: each vertex v_i with score $\mu_{i,r} > \tau$ is considered as being in class r. We move τ in steps through the complete range of values in $\boldsymbol{s}_r$ in descending order. For each step we update our prediction and calculate precision and recall. PRBEP is reported when the values are equal. Finally, we macro-average over all possible labels.

While PRBEP has been used as the evaluation metric in previous work [11,12], it does not capture the performance of the common multi-class inference rule $\hat{y}_i = \arg\max_r \mu_{i,r}$ used also by TACO. We thus also report accuracy. However, graph-based transduction algorithms tend to create degenerate solutions, such as solutions for which all unlabeled vertices are classified as being in the most common single class. Therefore, we also report macro-averaged accuracy: we divide the complete test set into disjoint sets according to true labels, compute accuracy on each individual set and average. This increases the effect of prediction accuracy in rare classes and demotes degenerate solutions.

In addition to PRBEP and accuracy, we report Mean Reciprocal Rank (MRR), $\text{MRR} = (1/|Q|) \sum_{v_i \in Q} (1/r_i)$ where Q is the test set and r_i is the rank of the true label y_i within the score vector $\boldsymbol{\mu}_i$. This metric has been recently used for comparing several graph-based transductive algorithms on the task of acquiring class-instance pairs from text [13]. While accuracy is based on prediction alone, this metric favours both accurate prediction as well as a good rank for the true label in case of a mistake. As before, in addition to MRR we report also macro-averaged MRR, discouraging degenerate solutions.

Table 2. A comparison of empirical results for three algorithms on all data sets. Each reported result is an average over 20 randomly generated transduction sets. Left block results were obtained by tuning hyper-parameters using macro-averaged PRBEP, while results on the right block were tuned using macro-averaged accuracy (M-ACC).

		Optimized by PRBEP			Optimized by M-ACC		
		MAD	AM	TACO	MAD	AM	TACO
WebKB 48 labeled	PRBEP	65.5	64.5	**67.7**	54.9	61.2	**67.7**
	ACC	22.0	43.8	**72.5**	59.5	63.5	**72.5**
	M-ACC	26.0	31.6	**70.9**	62.2	56.7	**71.5**
	MRR	49.8	66.1	**84.5**	76.5	78.7	**84.6**
	M-MRR	52.7	57.1	**83.4**	78.0	73.9	**83.9**
20 News 105 labeled	PRBEP	50.7	49.8	**57.0**	49.3	47.7	**55.6**
	ACC	16.6	40.3	**58.8**	50.0	45.8	**61.0**
	M-ACC	16.6	39.9	**57.9**	49.6	45.3	**60.2**
	MRR	34.9	54.5	**72.9**	65.3	59.8	**74.8**
	M-MRR	35.1	54.2	**72.4**	65.1	59.5	**74.4**
Sentiment 500 labeled	PRBEP	34.4	**34.9**	34.5	27.8	31.0	**33.7**
	ACC	24.3	**38.6**	25.4	32.6	**41.7**	38.3
	M-ACC	26.6	20.0	**27.4**	**32.5**	29.3	32.2
	MRR	50.7	**59.9**	51.6	56.9	**61.9**	60.7
	M-MRR	52.4	45.8	**53.1**	**56.6**	52.9	55.7
Reuters 48 labeled	PRBEP	**73.2**	**73.2**	73.0	72.0	67.6	**74.9**
	ACC	60.0	69.4	**82.7**	76.3	75.8	**81.6**
	M-ACC	59.1	55.6	**73.0**	75.6	67.2	**76.2**
	MRR	76.7	82.7	**90.0**	86.6	86.3	**89.7**
	M-MRR	76.2	72.8	**83.3**	**85.9**	80.1	**85.9**
Enron A 48 labeled	PRBEP	54.3	**54.8**	54.2	46.7	50.5	**50.9**
	ACC	45.4	**60.0**	46.0	49.5	**56.6**	55.6
	M-ACC	51.3	41.1	**52.0**	**52.4**	49.0	50.0
	MRR	62.8	**74.2**	63.3	66.2	70.2	**71.6**
	M-MRR	66.1	59.4	**66.7**	**67.0**	64.1	66.3
Enron B 48 labeled	PRBEP	39.7	36.3	**41.7**	36.7	35.7	**41.5**
	ACC	20.4	26.4	**41.4**	36.5	35.8	**41.1**
	M-ACC	21.0	16.6	**38.5**	**38.7**	29.9	38.4
	MRR	40.8	46.0	**59.1**	54.9	54.2	**58.7**
	M-MRR	39.3	36.4	**56.6**	55.5	48.4	**56.6**
Amazon3 35 labeled	PRBEP	**89.4**	77.7	88.4	75.4	74.1	**87.8**
	ACC	34.7	43.9	**88.5**	73.0	68.9	**88.9**
	M-ACC	34.8	43.9	**88.5**	73.0	68.9	**88.9**
	MRR	62.7	67.4	**93.9**	85.2	83.0	**94.2**
	M-MRR	62.8	67.4	**93.9**	85.2	83.0	**94.2**



(a) PRBEP tuned according to PRBEP (b) PRBEP tuned according to M-ACC

(c) M-ACC tuned according to M-ACC (d) M-ACC tuned according to PRBEP

Fig. 3. A comparison of macro-averaged PRBEP and macro-averaged accuracy (M-ACC) for different amounts of labeled data on the WebKB data set. All results are averages over 20 randomly generated transduction sets. In figures (a) and (d) hyper-parameters were tuned according to PRBEP. In figures (b) and (c) tuning is according to M-ACC. Error bars indicate 95% confidence intervals.

4.4 Results

Results for seven NLP tasks are summarized in Table 2. All results are averages over 20 randomly generated transduction sets as described in Sec. 4.2. We performed hyper-parameters tuning according to PRBEP since it is the reported metric in previous work [7,8,11,12]. In addition, we tune by M-ACC since it better relates to our inference rule and also penalizes degenerate solutions.

Results reported in the left part of Table 2 are from experiments where tuning was performed according to PRBEP. Focusing on the tuned metric, TACO outperforms the other two algorithms on three of the seven data sets by at least 2 points and is worse by at most 1 point. However, TACO outperforms the other algorithms on all datasets when evaluated using either M-ACC or M-MRR. This implies solutions generated by TACO are preferable.

Results reported in the right part of Table 2 were obtained by tuning with macro-averaged accuracy (M-ACC). Here, considering the tuned metric, TACO is best on four of the seven data sets, three of which by at least 9 points. When

TACO is not the best performing algorithm, the second best algorithm has M-ACC not larger than 1 point, except in one case where the difference is about 2.5 points. As before, considering other metrics, TACO is best on at least four data sets for all other metrics.

Comparing both parts of Table 2 suggests TACO is robust to the choice of the metric used for tuning. For example, on WebKB, tuning according to PRBEP or M-ACC does not change much the reported results. AM and MAD are more sensitive to the specific choice of a metric for tuning. Considering again WebKB, there are significant differences between results from the left and right parts of Table 2. In general, for most data sets, the difference between results in experiments tuned according to PRBEP or M-ACC is substantially smaller for TACO compared to both MAD and AM.

Overall performance is high in absolute values on WebKB, 20 Newsgroups, Reuters and Amazon3. On these data sets, TACO is best by almost any used evaluation metric. This suggests our algorithm gains from the merits of a good input graph better than others. Finally, comparing ACC vs M-ACC, and MRR vs M-MRR, we note that using averaged metrics the results are in general higher than using their macro-average counterparts. Yet, for TACO this difference is smaller, which indicates that TACO predicts better the less frequent classes, while not harming performance on the more frequent classes (see also Table 3).

Table 3. Precision-Recall Break Even Point (PRBEP) results per class on WebKB data set, with 48 labeled examples. All results are averages over 20 randomly generated transduction sets.

class	size	MAD	AM	TACO
course	930	**85.4**	81.1	**85.4**
faculty	1,124	58.9	59.9	**60.8**
project	504	41.4	42.6	**46.2**
student	1,641	76.1	74.4	**78.4**
average	-	65.5	64.5	**67.7**

A comparison of PRBEP and M-ACC vs amount of labeled examples is given in Fig. 3. In Fig. 3(a) and Fig. 3(c) the evaluation metric used for tuning is also the reported metric. In these two plots, we observe that performance increases as more labeled examples are given as input, as expected. TACO outperforms other algorithms, especially when the amount of labeled examples is small. Fig. 3(d) and Fig. 3(b) show how tuning by one metric affects the performance of the other metric. After tuning using PRBEP, TACO achieves far better M-ACC score than other algorithms, and vice-versa, although with smaller gap, tuning with M-ACC, TACO achieves better PRBEP score than others algorithms. Additionally, as more labeled examples are used, performance in the not-tuned metric may not grow. This implies tuning by PRBEP overfits the algorithm output towards solutions specifically maximizing the PRBEP measure, and results evaluated with other evaluation metrics such as M-ACC, are decreased. This phenomenon is most visible considering AM. As illustrated by Fig. 3(b), as the number of input labeled examples grows, performance in the not-tuned metric, namely PRBEP, increases. This suggests tuning by M-ACC is better for the WebKB data set.

Finally, per-class PRBEP for the WebKB dataset is summarized in Table 3. The average PRBEP (bottom line) corresponds to the top-left line in Table 2. TACO achieves the best performance in all four labels, and there is no clear winner between the two other algorithms. The highest improvement is for the *project* category, the one with the smallest number of test samples, (from 42.6 to 46.2), for which all algorithms obtain lowest performance among the four categories. It seems that the tuning via confidence information allows TACO to automatically adapt to unbalanced data.

5 Related Work

Our work builds on previous graph-based transduction learning, where the graph is built using nearest-neighbours based on some distance. Label propagation (LP) [14] and Modified Adsorption (MAD) [12] use squared Euclidean distance, while alternating minimization (AM) [10,11] uses the Kullback-Leibler divergence. To the best of our knowledge, our work is the first to apply second order information into transductive learning, implemented using the Mahalanobis distance with parameters (matrix) being optimized as well.

One of the first and best performing graph-based transduction algorithm based on ℓ_2 norm is label propagation (LP) [14], with update rule,

$$\mu_{i,r}^{(t)} = \delta_l(i) y_{i,r} + (1 - \delta_l(i)) \frac{\sum_{j=1}^{n} w_{i,j} \mu_{j,r}^{(t-1)}}{\sum_{j=1}^{n} w_{i,j}} \ .$$

Setting our confidence parameters $\sigma_{i,r}$ to 2, our update (12) becomes

$$\mu_{i,r}^{(t)} = \frac{\sum_{j=1}^{n} w_{i,j} \mu_{j,r}^{(t-1)} + C \cdot \delta_l(i) y_{i,r}}{\sum_{j=1}^{n} w_{i,j} + C \cdot \delta_l(i)} \qquad (14)$$

where we set $C = 1/2 + 1/\gamma$ and assume $w_{i,i} = 0$. The update step in (14) is a close variant of LP, allowing label information for labeled vertices to differ from the given known input labels. A very close version appears in a recent book [3] (Section 11.2, Algorithm 11.2). Thus, we can view this variant of LP as a special case of our algorithm with constant confidence parameters.

The idea of discouraging high degree vertices in label propagation first appears in Adsorption [1] and is later used in MAD [12]. Both algorithms use a static measure that considers only vertex degree to limit the effect of vertices with a large degree. TACO performs similar tuning, but automatically or dynamically based on the confidence parameters, which measure both degree and agreement level among neighbours.

Measures of confidence in estimated parameters have been successfully used in a number of non-SSL settings such as online learning [2,6] and multi-armed bandits with partial feedback [5].

6 Conclusion

We introduced the notion of confidence in label assignments to graph-based transduction. We formulated learning as an unconstrained convex optimization problem in both confidence and label parameters, and derived an efficient iterative algorithm for solving it. Our algorithm uses its confidence parameters to dynamically control the influence each vertex has on its neighbours during label propagation. Empirical evaluations on seven NLP tasks demonstrate that TACO improves over current state-of-the-art graph-based transductive algorithms, and is more robust to parameter tuning.

Currently, we plan to analyze TACO formally, experiment with more large-scale data and extend our work to multi-label, complex and structured problems. We also plan to generalize TACO to be able to generate models that will allow labeling unseen inputs beyond the transduction setting.

References

1. Baluja, S., Seth, R., Sivakumar, D., Jing, Y., Yagnik, J., Kumar, S., Ravich, D., Aly, R.M.: Video suggestion and discovery for youtube: taking random walks through the view graph. In: WWW (2008)
2. Cesa-Bianchi, N., Conconi, A., Gentile, C.: A Second-Order Perceptron Algorithm. SIAM Journal on Computing 34(3), 640 (2005)
3. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge (2006)
4. Crammer, K., Dredze, M., Kulesza, A.: Multi-class confidence weighted algorithms. In: EMNLP (2009)
5. Crammer, K., Gentile, C.: Multiclass classification with bandit feedback using adaptive regularization. In: ICML, pp. 273–280 (2011)
6. Dredze, M., Crammer, K.: Confidence-weighted linear classification. In: ICML, pp. 264–271 (2008)
7. Joachims, T.: Transductive inference for text classification using support vector machines. In: ICML, pp. 200–209 (1999)
8. Joachims, T.: Transductive learning via spectral graph partitioning. In: ICML, pp. 290–297 (2003)
9. Lewis, D.D., Yang, Y., Rose, T.G., Li, F., Dietterich, T.G.: Rcv1: A new benchmark collection for text categorization research. In: JMLR (2004)
10. Subramanya, A., Bilmes, J.: Soft-supervised learning for text classification. In: EMNLP (2008)
11. Subramanya, A., Bilmes, J.: Semi-Supervised Learning with Measure Propagation. The Journal of Machine Learning Research 12, 3311–3370 (2011)
12. Talukdar, P.P., Crammer, K.: New Regularized Algorithms for Transductive Learning. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 442–457. Springer, Heidelberg (2009)
13. Talukdar, P.P., Pereira, F.: Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In: ACL (2010)
14. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML (2003)

Maximum Consistency Preferential Random Walks

Deguang Kong and Chris Ding

Dept. of Computer Science & Engineering, University of Texas at Arlington, TX, 76013
doogkong@gmail.com, chqding@uta.edu

Abstract. Random walk plays a significant role in computer science. The popular PageRank algorithm uses random walk. Personalized random walks force random walk to "personalized views" of the graph according to users' preferences. In this paper, we show the close relations between different preferential random walks and label propagation methods used in semi-supervised learning. We further present a maximum consistency algorithm on these preferential random walk/label propagation methods to ensure maximum consistency from labeled data to unlabeled data. Extensive experimental results on 9 datasets provide performance comparisons of different preferential random walks/label propagation methods. They also indicate that the proposed maximum consistency algorithm clearly improves the classification accuracy over existing methods.

1 Introduction

Random walk model [1] is a mathematical formalization of the paths that consist of taking successive random steps, i.e., at each step the walk jumps to another site according to some probability distribution. The random walk model plays an important role in computer science, and it has many applications in information retrieval [2], social network [3], etc. PageRank [4] is a link analysis algorithm, which uses the idea of random walk to measure the webpages' relative importances. Personalized Page Rank [5] is presented to create "personalized views" of the web searching results based on redefining importances according to users' preferences.

Semi-supervised learning(SSL) has connections with random walks on graphs. In SSL, only a small number of data points are labeled while a large number of data points are unlabeled. The goal of SSL is to classify the unlabeled data based on labeled data. SSL has attracted more attention because the acquisition of labeled data is quite expensive and time-consuming, while large amount of unlabeled data are easier to obtain. Many different methods have been proposed to solve SSL problems [6,7], e.g., classification-based method [8], clustering-based method [9], graph-based method [10,11,12], etc. Among all these methods, the graph-based method is the most popular way to model the whole dataset as undirected weighted graph with pairwise similarities($\mathbf{W}$), and the semi-supervised learning can be viewed as label propagation from labeled data to unlabeled data, like a random walk on a similarity-graph $\mathbf{W}$. Our work is inspired by previous graph-based semi-supervised methods, especially by the works of consistency labeling [11] and Green's function [12].

In this paper, we *first* show the close relations between preferential random walks and label propagation methods. We show that the labeled data points act as the preferential/personalized bias vectors in the personalized random walks. This provides much

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 339–354, 2012.

insight to the existing label propagation methods, and suggest ways to improve these methods. In addition, we perform extensive experiments to compare the performances of different methods used in preferential random walks.

Furthermore, we observe that current label-propagation approach may not achieve best available results, especially when the propagation operator, inferred from both labeled and unlabeled data points, does not exactly reveal the intrinsic structure of data. Many label propagation methods are done in one shot from source (labeled data) to all unlabeled data. This can not guarantee many newly-labeled data, which lie far-away in the data manifold of both labeled and unlabeled data, are labeled reliably. Motivated by this observation, in this paper, we present a novel maximum consistency approach to improve the performance of existing label propagation methods. Our approach focuses on propagating labels from source to *nearby* unlabeled data points only, and thus reliably labeling these data points. This propagation expands progressively to all unlabeled data, to ensure maximum consistency from labeled data to unlabeled data. Maximum consistency algorithm leverages existing propagation methods and repeatedly utilizes it, which incurs almost the same computational complexity as other existing propagation methods.

Here we summarize the contribution of our paper.

- We show the direct relations between preferential random walks and existing label propagation methods. Extensive experiments on 9 datasets are performed to demonstrate the performance of different methods.
- We present a maximum consistency algorithm to improve existing label-propagation methods. Extensive experiments performed on 9 datasets indicate clear performance improvement.

The rest of this paper is organized as follows. §2 gives a brief overview of personalized random walk. Next in §3, we establish the connections between the preferential random walks and label propagation methods. In §4, we emphasize the concept of score distribution in semi-supervised learning methods. In §5, we propose maximum consistency label propagation method. §6 reviews the related work to our paper. In §7, extensive experiments on 9 datasets are performed to provide the performance comparisons of both different preferential random walks/label propagation methods and proposed maximum consistency algorithm. Finally, we conclude the paper.

2 A Brief View of Personalized Random Walk

On an undirected graph with edge weights $\mathbf{W}$, let $\mathbf{D}$ be the diagonal matrix with $\mathbf{D} = \mathrm{diag}(\mathbf{We}), \mathbf{e} = (1, \ldots, 1)^T$, then $\mathbf{P} = (\mathbf{P}_{ij})$ is the transition probability from node i to node j,

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W} \tag{1}$$

Let $\mathbf{f}_i$ be the stationary probability of one random walker on site i. The following propagation

$$\mathbf{f} = (1 - \alpha)\mathbf{y} + \alpha\mathbf{P}^T\mathbf{f}, \tag{2}$$

governs the random walker. The converged stationary distribution gives the score.

Here $\mathbf{y}$ is the personalized (bias) probability distribution; this fixed vector represents the personal interest or other preferential treat of different nodes. In PageRank [4], $\mathbf{y} = (1, \ldots, 1)^T/n$, $\alpha = 0.9$. In personalized random walk [5], $\mathbf{y}$ encodes the personalized preferences. For example, for a random walker who prefers to visit sites i_1, i_2. Then $\mathbf{y}_i = 1/2$ if $i = i_1, i_2$; $\mathbf{y}_i = 0$ otherwise.

2.1 Personalized Random Walk for 2-Class Semi-supervise Learning

To do classification for partially labeled data for 2-class, we divide the data into $\mathbf{X}_+$, $\mathbf{X}_-$, and $\mathbf{X}_u$ for positively labeled, negatively labeled, and unlabeled datasets. We do two random walks: (1) one for the positive class with preferential vector $\mathbf{y}^{(+)}$ where $\mathbf{y}_i^{(+)} = 1/|\mathbf{X}_+|$ if $i \in \mathbf{X}_+$; $\mathbf{y}_i^{(+)} = 0$ otherwise. The converged score of Eq.(2) gives $\mathbf{f}^{(+)}$. (2) one for the negative class with preferential vector $\mathbf{y}^{(-)}$ where $\mathbf{y}_i^{(-)} = 1/|\mathbf{X}_-|$ if $i \in \mathbf{X}_-$; $\mathbf{y}_i^{(-)} = 0$ otherwise. The converged score of Eq.(2) gives $\mathbf{f}^{(-)}$. We then assign for each unlabeled data $\mathbf{x}_i \in \mathbf{X}_u$ the class with higher stationary distribution: $k = \max(\mathbf{f}_i^{(+)}, \mathbf{f}_i^{(-)})$.

Note that because the propagation of Eq.(2) is linear, we can do the semi-supervised learning using only *one* random walk with the preferential vector $\mathbf{y} = \frac{1}{2}(\mathbf{f}^{(+)} - \mathbf{f}^{(-)})$. We then assign for each unlabeled data $\mathbf{x}_i$ the class with $k = \text{sign}(\mathbf{f}_i)$. This is a simple algorithm. Note that here $\sum_i \mathbf{y}_i = 0$, since $\sum_i \mathbf{y}_i^{(+)} = 1$ and $\sum_i \mathbf{y}_i^{(-)} = 1$. This will be useful in deriving the Green's function method below.

2.2 Generalized Preferential Random Walk for Multi-class

In multi-person random walks, there are K random walkers. Each random walker $k(1 \leq k \leq K)$has a distribution vector $\mathbf{f}_k$ and a personalized preference vector $\mathbf{y}_k$,

$$\mathbf{f}_k = (1-\alpha)\mathbf{y}_K + \alpha \mathbf{P}^T \mathbf{f}_K. \tag{3}$$

Let $\mathbf{F} = (\mathbf{f}_1, \cdots, \mathbf{f}_K)$ and $\mathbf{Y} = (\mathbf{y}_1, \cdots, \mathbf{y}_K)$, from Eq.(2), we obtain the transition

$$\mathbf{F} = (1-\alpha)\mathbf{Y} + \alpha \mathbf{P}^T \mathbf{F}. \tag{4}$$

The solution for the final stationary distributions of the K random walkers are

$$\mathbf{F} = \frac{1-\alpha}{\mathbf{I} - \alpha \mathbf{P}^T}\mathbf{Y}. \tag{5}$$

Method 1:
Here we use standard random walk transition probability of Eq.(1) and obtain the stationary distributions of the K random walkers:

$$\mathbf{F} = \frac{1-\alpha}{\mathbf{I} - \alpha \mathbf{W}\mathbf{D}^{-1}}\mathbf{Y} = \frac{1-\alpha}{(\mathbf{D} - \alpha \mathbf{W})\mathbf{D}^{-1}}\mathbf{Y} = \mathbf{D}\frac{1-\alpha}{(\mathbf{D} - \alpha \mathbf{W})}\mathbf{Y}. \tag{6}$$

Method 2:
If we use the "pseudo transition probability" $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, we obtain the stationary distributions of the K random walkers as:

$$\mathbf{F} = \frac{1-\alpha}{\mathbf{I} - \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}}\mathbf{Y}. \tag{7}$$

Method3:
If we use another "pseudo transition probability" $\mathbf{P} = \mathbf{W}\mathbf{D}^{-1}$, we obtain the stationary distributions of the K random walkers as:

$$\mathbf{F} = \frac{1-\alpha}{\mathbf{I} - \alpha\mathbf{D}^{-1}\mathbf{W}}\mathbf{Y} = \frac{1-\alpha}{\mathbf{D}^{-1}(\mathbf{D} - \alpha\mathbf{W})}\mathbf{Y} = \frac{1-\alpha}{(\mathbf{D} - \alpha\mathbf{W})}\mathbf{D}\mathbf{Y}. \tag{8}$$

So far, we have discussed random walks on a graph. Next, we make connections to semi-supervised learning. The significance of relation analysis between preferential random walks and label propagations is to help to capture the essence of these algorithms and better interpret the experiment results. To our knowledge, so far there is a lack of systematic study to explore the commonalities and differences of these algorithms, and their relations to label propagation algorithms.

3 Relations between Preferential Random Walks and Label Propagations

In semi-supervised learning, we have $n = n_\ell + n_u$ data points $\{\mathbf{x}_i\}_{i=1}^{n}$, where first n_ℓ data points are already labeled with $\{y_i\}_{i=1}^{n_\ell}$ for c target classes. Here, $\mathbf{x}_i \in \Re^p$ and $y_i \in \{1, 2, ..., K\}$, such that $y_i = k$ if x_i belongs to the k-th class. The last n_u data are unlabeled. The goal of semi-supervised learning is to learn their class labels: $\{y_i\}_{i=n_\ell+1}^{n}$. Let $\mathbf{Y} \in \Re^{n\times K}$ be a class indicator matrix, $\mathbf{Y}_{ij} = 1$ if $\mathbf{x}_i$ is labeled as class $y_i = j$; and $\mathbf{Y}_{ij} = 0$ otherwise.

3.1 Local - Global Consistency Method(LGC)

Local and global consistency(LGC) [13] utilizes sufficiently smooth assumptions with respect to the intrinsic structure collectively revealed by known labeled and unlabeled data points. Given the graph edge matrix $\mathbf{W}$, LGC constructs the normalized matrix $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, where D is a diagonal matrix with $\mathbf{D} = \mathrm{diag}(\mathbf{W}\mathbf{e})$. Then the predicted label matrix $\mathbf{F}$ is,

$$\mathbf{F} = \mathbf{Q}\mathbf{Y}, \qquad \mathbf{Q} = \beta(\mathbf{I} - \alpha\mathbf{S})^{-1}, \tag{9}$$

where $\mathbf{Q}$ is the label propagation operator, $\alpha = \frac{1}{1+\mu}$, $\beta = \frac{\mu}{1+\mu}$, and μ is a parameter.

Relations with Preferential Random Walk. Compared with method 2 in generalized preferential random walk of Eq.(7), we can see LGC is *identical* to it. This is because constant β will not change the classification results.

3.2 Green's Function Method(GF)

Green's function for semi-supervised learning and label propagation is first presented in [12]. GF is defined as the inverse of graph laplacian $\mathcal{L} = \mathbf{D} - \mathbf{W}$ with zero-mode discarded. Using the eigenvectors of $\mathcal{L}$: $\mathcal{L}\mathbf{v}_k = \lambda_k \mathbf{v}_k$, where $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are the eigenvalues. Green's function computes the predicted label matrix $\mathbf{F}$,

$$\mathbf{F} = \mathbf{QY}, \qquad \mathbf{Q} = \mathcal{L}_+^{-1} = \frac{1}{(\mathbf{D} - \mathbf{W})_+} = \sum_{i=2}^{n} \frac{\mathbf{v}_i \mathbf{v}_i^T}{\lambda_i}, \tag{10}$$

where $\mathbf{Q}$ is label propagation operator, $(\mathbf{D} - \mathbf{W})_+$ indicates zero eigen-mode is discarded.

Relations with Preferential Random Walk. From Method 1 of generalized preferential random walk, the stationary distribution $\mathbf{F}$ of Eq.(6) is related to Q in Eq.(10). As $\alpha \longrightarrow 1$, we have

$$(\mathbf{D} - \alpha\mathbf{W})^{-1} \longrightarrow (\mathbf{D} - \alpha\mathbf{W})^{+} = \sum_{i=2}^{n} \frac{\mathbf{v}_i \mathbf{v}_i^T}{\lambda_i}. \tag{11}$$

Indeed, for classification purpose, the GF approach is the limit of Method 1 of generalized preferential random walk of Eq.(6). This is further explained below:

(1) In semi-supervised learning, the classification result for object i is determined by the location of the largest element in i-th row(See Eq.12).

(2) Given a distribution $\mathbf{A}$ and a diagonal matrix $\mathbf{D} = \mathrm{diag}(d_1 \cdots d_n)$, $\mathbf{DA}$ will multiply the i-th row of $\mathbf{A}$ by d_i. The relative distribution of this row does not change. Thus $\mathbf{D}$ applied to distribution $\mathbf{A}$ does not change the classification results.

(3) The multiplicative constant $(1 - \alpha)$ does not change the classification too.

(4) The physical reason of discarding zero mode is the use of the Von Neumann boundary condition. Algorithmically, this is also consistent: First, the discarded zero mode in Eq.(11) is $\mathbf{v}_1\mathbf{v}_1^T/\lambda_1 = \mathbf{e}\mathbf{e}^T/(n\lambda_1)$ where $\lambda_1 = 0$. As discussed in §2.1, the multi-class random walk can be equivalently viewed as a single random walk with preference vector $\mathbf{y} = \frac{1}{2}(\mathbf{y}^{(k)} - \mathbf{y}^{(\bar{k})})$, where $\mathbf{y}^{(k)}$ is the preference vector for class k, and $\mathbf{y}^{(\bar{k})}$ is the preference vector for other classes $\bar{k}$. Note $\sum_i \mathbf{y}_i = 0$, since $\sum_i \mathbf{y}_i^{(k)} = 1$ and $\sum_i \mathbf{y}_i^{(\bar{k})} = 1$. Thus we have $(\mathbf{v}_1\mathbf{v}_1^T/\lambda_1)\mathbf{y} = 0$, indicating including the zero mode in Eq.(11) does not alter the final results of label propagation.

3.3 Comparison of Preferential Random Walk Results

In label propagation of Eq.(9) or Eq.(10), once the distribution score (a.k.a propagation score) $\mathbf{F}$ are obtained, each unlabeled data point $\mathbf{x}_i$ is assign a class label according to

$$k = \arg\max_{1 \leq j \leq c} \mathbf{F}_{ij} \tag{12}$$

Note the key difference of LGC with GF is the computation of propagation operator $\mathbf{Q}$: LGC uses Eq.(9) while GF uses Eq.(10), which leads to different label propagation results. Another popular label propagation method is Harmonic function [10], which emphasizes harmonic nature of the label diffusive process. It is very different from LGC and Green's function, thus we did not discuss it here.

We have done extensive experiments to compare the above discussed methods for semi-supervised learning. We defer the presentation of these results in the experiment §7. We next discuss another contribution of this paper, i.e., the maximum consistency algorithm on these preferential random walk/label propagation methods.

4 Score Distribution: Confidence of Label Assignment

We begin the presentation of our maximum consistency with analysis of the distribution scores of the propagation. Our approach is motivated by careful examinations of experiment results. One conclusion is that although label propagation methods are effective, their current achieved results can be improved significantly. Below we illustrate the reasons.

In both LGC (Eq.9) and GF (Eq.10) methods, the propagation is done in one shot. All unlabeled data obtain their class labels immediately. However, some unlabeled data points may lie near labeled data in the data manifold (embedding subspace), while many other unlabeled data lie far-away from the labeled data. Therefore, the **reliability** or confidence of the class labels obtained in propagation vary from high (for those lie near labeled data) to low (for those lie far-away from labeled data).

However, in the *currently standard* class assignment procedure of Eq.(12), the class decision is simply the largest one among the c classes in the propagation score distribution across c classes. For example, for $\mathbf{x}_i$, the score distribution maybe

$$(\mathbf{F}_{i1}\cdots\mathbf{F}_{ic}) = (0.1,\ 0.2,\ 0.8,\ 0.3,\ 0.05),$$

in a data with $c = 5$ classes. For $\mathbf{x}_j$, the score distribution maybe

$$(\mathbf{F}_{j1}\cdots\mathbf{F}_{jc}) = (0.2,\ 0.35,\ 0.38,\ 0.05,\ 0.3).$$

Even though both $\mathbf{x}_i, \mathbf{x}_j$ are assigned class label=3, the confidence of the assignments are different. Clearly, $\mathbf{x}_i$ is assigned with higher confidence because $\mathbf{F}_{i3} = 0.8$ is much higher than other classes. $\mathbf{x}_j$ is assigned with lower confidence because $\mathbf{F}_{j3} = 0.38$ is marginally higher than some other classes. In other words, for $\mathbf{x}_i$ the propagation score distribution has a sharp peak while for $\mathbf{x}_j$ the propagation score distribution has a rather flat peak.

There could be many reasons that $\mathbf{x}_i$'s score distribution is much sharper than the score distribution for $\mathbf{x}_j$. $\mathbf{x}_i$ could lie much closer to class$=3$ labeled data point than $\mathbf{x}_j$. It could also be that there are more class$=3$ labeled data near $\mathbf{x}_i$ than near $\mathbf{x}_j$. It is also possible that there are many unlabeled points near $\mathbf{x}_i$ such that they mutually enhance the class$=3$ probability than those near $\mathbf{x}_j$. More possibilities exist. Fortunately, it is not necessary to dig out these details — they are *collectively* reflected in the propagation score distribution.

In existing label propagation approaches, both $\mathbf{x}_i, \mathbf{x}_j$ are assigned labels in one shot. Now consider a different approach where we break the actual label assignment into several rounds. We first assign class label for $\mathbf{x}_i$ and move it to the pool of already-labeled data, while defer the decision for $\mathbf{x}_j$ in later rounds. As the pool of already-labeled data expands to the neighborhood of $\mathbf{x}_j$, the propagation score distribution for

$\mathbf{x}_j$ is likely to become sharper. At this time/round, we assign class label to $\mathbf{x}_j$. Thus the class label assignment is always occurring at the situation where the assignment is done with high confidences, i.e., the assignment is done such that the data point is the most consistent with other members of the same class, both globally and locally, as reflected by the sharp score distribution. From these observations and discussions, we design a maximum consistent(MC) label propagation algorithm, which uses the label propagation operator $\mathbf{Q}$ defined in both LGC and GF methods. We call our approach as MC-LGC and MC-GF. Detailed algorithm is presented in next section.

5 Maximum Consistency Label Propagation

5.1 Design of the Algorithm

Our algorithm design is guided by maximum consistency assumption, which consists of multiple label propagations,

$$\begin{aligned} \mathbf{F}^1 &= \mathbf{QY}^0, \\ \mathbf{F}^2 &= \mathbf{QY}^1, \\ &\cdots \\ \mathbf{F}^t &= \mathbf{QY}^{t-1}, \end{aligned} \tag{13}$$

where $\mathbf{Q}$ is the propagation operator which can be computed from Eq.(9) or Eq.(10), and $\mathbf{F}^t$ is the label prediction matrix during each propagation. In each label propagation process, we use the current labeled data matrix $\mathbf{Y}^t$ to update the label prediction matrix $\mathbf{F}^t$.

At the end of each propagation, only those unlabeled data points whose class labels are reliably predicted are actually assigned class labels and moved into the pool of labeled data (Lpool). The rest of unlabeled data points remain in the pool of unlabeled data (Upool). Thus the pool of unlabeled data decreases with each propagation, and the pool of labeled data expands with each propagation. At last propagation, all remaining unlabeled data are assigned class labels.

Because of class balance consideration, the pool of labeled data should get approximately the same number of new members for each class. In our algorithm, each class gets one new member after each propagation. We call this procedure as “balanced class expansion (BCE)”. The number of unlabeled data are shrinking while the number of labeled data are increasing during this repeated BCE procedure. The critical issue in this BCE procedure is how to select this new member for each class. i.e., how to decide “reliably predicted” data points in each BCE. As analyzed in above section, the reliability of label propagation is reflected in score distribution. Thus, in our algorithm, we use the score distribution to decide the most “reliable predicted” data points from the data points in Upool in each BCE. We will illustrate more details in the next section.

Discussion. If we add different number of new members to different classes, it will produce unbalance. Even if the discriminant scores of one class are much higher than those of another class, we still consider add one number for each class. Although it is inefficient, we believe this conservative way will result in selection of more “reliable” data points.

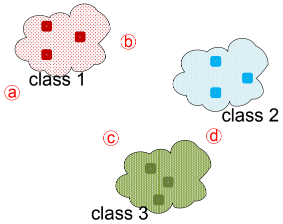


Fig. 1. Selection of discriminative data in balanced class expansion. Data points: a, b, c, d.

5.2 Normalization on the Distribution Score

Although data in Lpool expands in a class-balanced way, there are always the situation where classes become unbalanced. In the label propagation, we need to properly normalize the contributions from each class.

Suppose, a subset of data are labeled and there exists a class prior probability π_k. Let $\pi = \text{diag}(\pi_1 \cdots \pi_k)$, and $\mathbf{Z}$ be the multi-class label assignment matrix from labeled data, i.e.,

$$\mathbf{Z}_{ik} = \begin{cases} 1, & \text{if } \mathbf{x}_i \text{ belongs to class k} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

then the balanced source of propagation is defined as

$$\mathbf{Y} = \mathbf{Z}\pi = \begin{pmatrix} \pi_1\mathbf{Z}_{1,1} & \cdots & \pi_c\mathbf{Z}_{1,c} \\ \cdots & \cdots & \cdots \\ \pi_1\mathbf{Z}_{n,1} & \cdots & \pi_c\mathbf{Z}_{n,c} \end{pmatrix}. \tag{15}$$

In our algorithm, we set the prior to $\pi_k = \frac{1}{\sum_i \mathbf{Z}_{ik}}$. therefore, each class contributes the same total weight to the propagation: $\sum_i \mathbf{Y}_{ik} = \sum_i \mathbf{Y}_{i\ell}$ for any two class k, ℓ. In our algorithm the initial label matrix $\mathbf{Y}^0$ is constructed as

$$\mathbf{Y}^0 = \mathbf{Z}^0\pi^0, \tag{16}$$

where $\mathbf{Z}^0$ is the initial label assignment matrix constructed as Eq.(14) from the initially labeled data in Lpool. In the t-th iteration, let $\mathbf{Z}^t$ be the label assignment matrix constructed from current data in Lpool,

$$\mathbf{Y}^t = \mathbf{Z}^t\pi^t. \tag{17}$$

5.3 Reliable Assigning Class Labels with Score Distribution

After obtaining the assignment score $\mathbf{F}_{ik}$ for all data in Upool, our goal is to pick up the "reliable" assigned data points, one for each class, and add them to the Lpool whereas remove them from the Upool. Afterwards in the actual label assignment for each class, we (1) find out all the currently unlabeled data assigned to this class, (2) pick the one with the highest discriminative score and assign it to this class.

A Motivating Example to Illustrate Discriminant Score. Fig.(1) illustrates the idea of selecting discriminative unlabeled data points. Class 1 selects data a instead of data

b, because a is far away from classes 2 and 3; although b is slightly closer to class 1, but b is also closer to class 2. In other words, a is more class discriminative than b. Similarly, class 2 selects data c instead of d, because c is more discriminative than d.

Now we discuss the discriminative score computation. For each unlabeled data point $\mathbf{x}_i$, it has been assigned to k scores($\mathbf{F}_{ik}, 1 \leq k \leq c$). The c scores are then sorted as,

$$\mathbf{F}_{ik_1} \geq \mathbf{F}_{ik_2} \geq \mathbf{F}_{ik_3} \geq ... \tag{18}$$

3 classes with the highest scores are recorded as the three closest classes for $\mathbf{x}_i$: $\mathbf{F}_{k_1}$; $\mathbf{F}_{k_2}$, $\mathbf{F}_{k_3}$. As discussed above, even two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ have been assigned to the same class c_k, they may have different discriminant scores depending on the scores which how $\mathbf{x}_i, \mathbf{x}_j$ may be assigned to other classes. Here we consider the **target** class the data points will be assigned to and other two **competing** classes which we wish to be discriminant against. The discriminative scores for the 1st choice target class are defined as (if there is only 2 classes, we do not need c_{k3}),

$$\mathbf{E}(i, c_{k1}) = \mathbf{F}^2_{ic_{k1}} \frac{|\mathbf{F}_{ic_{k1}} - \mathbf{F}_{ic_{k2}}| + |\mathbf{F}_{ic_{k1}} - \mathbf{F}_{ic_{k3}}|}{\sqrt{\mathbf{F}_{ic_{k1}} + \mathbf{F}_{ic_{k2}} + \mathbf{F}_{ic_{k3}}}}. \tag{19}$$

The score difference achieves the discriminative affects. The denominator provides a mild scale normalization. Without this term, the class with largest $\mathbf{F}_{ik}$ scale may dominate the score computation process. Note that these scores are computed once for each balanced class expansion. For each unlabeled data point $\mathbf{x}_i$ in Upool, it is assigned to class k, which has the largest $\mathbf{F}_{ik}$scores among all class k. For each class k, we select the data points $\mathbf{x}_i$, which has the largest discriminative score $\mathbf{E}(x_i, c_k)$ among all data points in Upool assigned to class k. This procedure is designed to maximize the label assignment consistency, which is consistent with LGC/GF approach.

Discussion on the Discriminant Score. Actually, we can define other formulations of discriminant score. (1) Without the denominator of Eq.(19), discriminant score can be written as,

$$\mathbf{E}_2(i, c_{k1}) = \mathbf{F}^2_{ic_{k1}}(|\mathbf{F}_{ic_{k1}} - \mathbf{F}_{ic_{k2}}| + |\mathbf{F}_{ic_{k1}} - \mathbf{F}_{ic_{k3}}|). \tag{20}$$

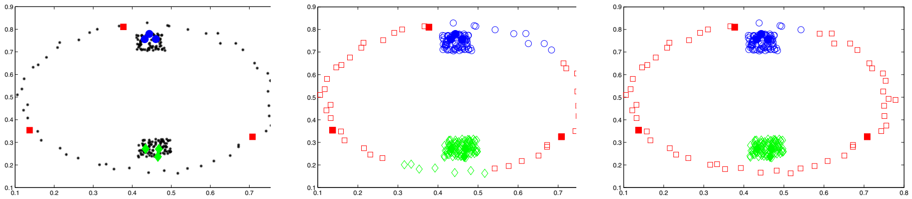
(2) Without the square for the 1st term of Eq.(19), discriminant score can be written as,

$$\mathbf{E}_3(i, c_{k1}) = \mathbf{F}_{ic_{k1}} \frac{|\mathbf{F}_{ic_{k1}} - \mathbf{F}_{ic_{k2}}| + |\mathbf{F}_{ic_{k1}} - \mathbf{F}_{ic_{k3}}|}{\sqrt{\mathbf{F}_{ic_{k1}} + \mathbf{F}_{ic_{k2}} + \mathbf{F}_{ic_{k3}}}}. \tag{21}$$

(3) Select more top (e.g., $4, 5, 6, 7, \cdot\cdot$) classes to compute the discriminant score, then discriminant score for T classes is given by,

$$\mathbf{E}_4(i, c_{k1}) = \mathbf{F}^2_{ic_{k1}} \frac{\sum_{t=1}^{T} |\mathbf{F}_{ic_{k1}} - \mathbf{F}_{ic_{kt}}|}{\sqrt{\sum_{t=1}^{T} \mathbf{F}_{ic_{kt}}}}. \tag{22}$$

Our experiments results(see §7.4) show Eq.(19) achieves slightly better results than other discriminant scores defined in Eqs.(20,21,22). For Eq.(20), the denominator is removed. When some $\mathbf{F}_{ic_k}$ has very large values, it may dominator the score. For Eq.(21), square of score $\mathbf{F}_{ic_k}$ is removed, which makes the score less sharper than that of Eq.(19). For Eq.(22), more top classes are fetched to achieve discriminant effect. In our experiments, we find when we select 3 classes, we can get very good results. When we select more classes, the results change slightly, but sometimes even worse.



(a) Data distribution.

(b) LGC result

(c) MC-LGC result

Fig. 2. Illustration of maximum consistency approach on a synthetic dataset. Labeled data shown in thick symbols: red squares, green diamonds, blue circles for 3 classes. Initially unlabeled data are shown in black stars and, after obtaining labels, shown in open symbols.

Algorithm 1. Maximum consistency label propagation algorithm (MC algorithm)

```
Input: labeled data $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$, unlabeled data $U = \{\mathbf{x}_j\}_{j=\ell+1}^{\ell+u}$, MaxIter
Output: predicted class labels for unlabeled data
Procedure:
1:  compute propagation operator $\mathbf{Q}$ with Eq.(9) or Eq.(10), compute initial label matrix $\mathbf{Y}^0$ using Eq.(16), $t = 1$
2:  while $t <$ MaxIter & $U$ is not empty do
3:      $\mathbf{F}^t = \mathbf{Q}\mathbf{Y}^{t-1}$
4:      for each unlabeled data do
5:          compute its corresponding discriminative score using Eq.(19)
6:      end for
7:      for $k = 1$ to $c$ do
8:          search all unlabeled data whose 1st choice target class is k. {Balanced class expansion}
9:          if not empty then
10:             pick the one with the largest discriminative score, add it to class k, remove it from $U$
11:         end if
12:     end for
13:     Update $\mathbf{Y}^t$ with Eq.(17) using current label assignment $\mathbf{Z}^t$ {new labeled data added to Lpool}
14:     $t = t + 1$
15: end while
```

Demonstration of Algorithm Performance on Toy Data. We illustrate the advantage of the MC approach (on LGC methods) in Fig.2. A 3-class synthetic dataset is displayed in Fig.(2a). For each class, three data points are labeled while the rest of data points are unlabeled. Results of standard LGC methods and MC-LGC methods are shown in Figs.(2b, 2c). It is clear that MC approaches achieves better results. One can get similar results if making the comparisons of GF against MC-GF methods.

Complete algorithm is listed in Algorithm 1. This algorithm wraps around the label propagation operator $\mathbf{Q}$, and it can also use other label propagation operators.

Time Complexity Analysis. Note we only need to compute propagation operator $\mathbf{Q}$ (through Eq.10 or Eq.9) once as in standard LGC or GF, and the extra time cost is the iteration cost in balanced class expansion(BCE) process, which includes (1) the iteration time of BCE process which is proportional to number of iteration t; (2) the discriminant score table computation in lines $7 - 13$ of Algorithm 1, which is proportional to the number of *current* unlabeled data points n_l and the number of class label c. In our experiment, we find that the extra time cost is very limited as compared to the propagation operator computation in step 1.

6 Related Works

Here we discuss the previous works related to our algorithm. The related methods can be roughly divided into three categories, (1) personalized random walk (RW); (2) semi-supervised learning(SSL); (3) belief propagation (BP).

Random Walk is a popular technique widely used for PageRank algorithm [4]. Many variations of random walk methods are proposed, including personalized page rank [5], lazy random walks [14], fast random walk with restart [15], center-piece sub-graph discovery [16], using ghost edge for classification [17], analysis [18] and so on.

Semi-Supervised Learning methods are widely used in real applications. Graph-based semi-supervised methods are the most popular and effective methods in semi-supervised learning. The key-idea of graph-based semi-supervised methods is to estimate a (label propagation) function on a graph, which maximizes (1) consistency with the label information; (2) the smoothness over the whole graph. Several representative methods include harmonic function [10], local and global consistency [11] and Green's function [12].

Belief Propagation [19] is widely used for inference in probability graphical model. Belief propagation methods can be used for collective classification for network data [20], grouping nodes into regions for graphs [21] and so on. However, the computational cost for BP method is usually very high.

Maximum consistency label propagation is an improvement of state-of-the-art semi-supervised learning methods, which can be extended for collective classification [20] and community detection [22]. Due to space limit, we omit the discussions here.

7 Experiments

In this section, we perform two groups of experiments. One group is to compare three different methods in preferential random walks of Eqs.(6-8), and the other group is to evaluate the effectiveness of maximum consistency (MC) algorithm.

7.1 Datasets

We adopt 9 data sets in our experiments, including two face datasets AT&T and umist, three digit datasets mnist [23], binalpha and digit[1], two image scene datasets Caltec101 [24,25] and MSRC [25], and two text datasets Newsgroup[2] and Reuters[3]. Table 1 summarizes the characteristics of the datasets.

7.2 Experiments Results on 3 Methods of Generalized Preferential Random Walks of Eqs.(6-8)

In §2, we give three methods for generalized preferential random walks. We show method 2 is equivalent to LGC method. When $\alpha = 0.1$, GF method is the limit of

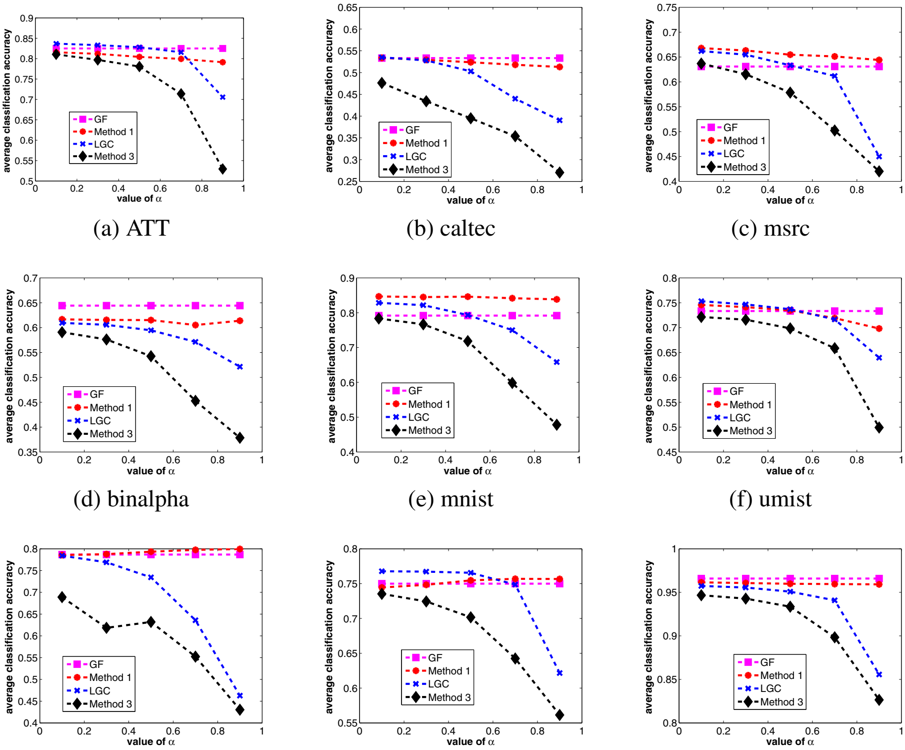
[1] http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html

[2] http://people.csail.mit.edu/jrennie/20Newsgroups/

[3] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Table 1. Descriptions of datasets

Dataset	#Size	#Dimension	#Class
AT&T	400	644	40
Caltech	600	432	20
MSRC	210	432	7
Binalpha	1014	320	36
Mnist	150	784	10
Umist	360	644	20
Newsgroup	499	500	5
Reuters	900	1000	10
digit	1500	241	2



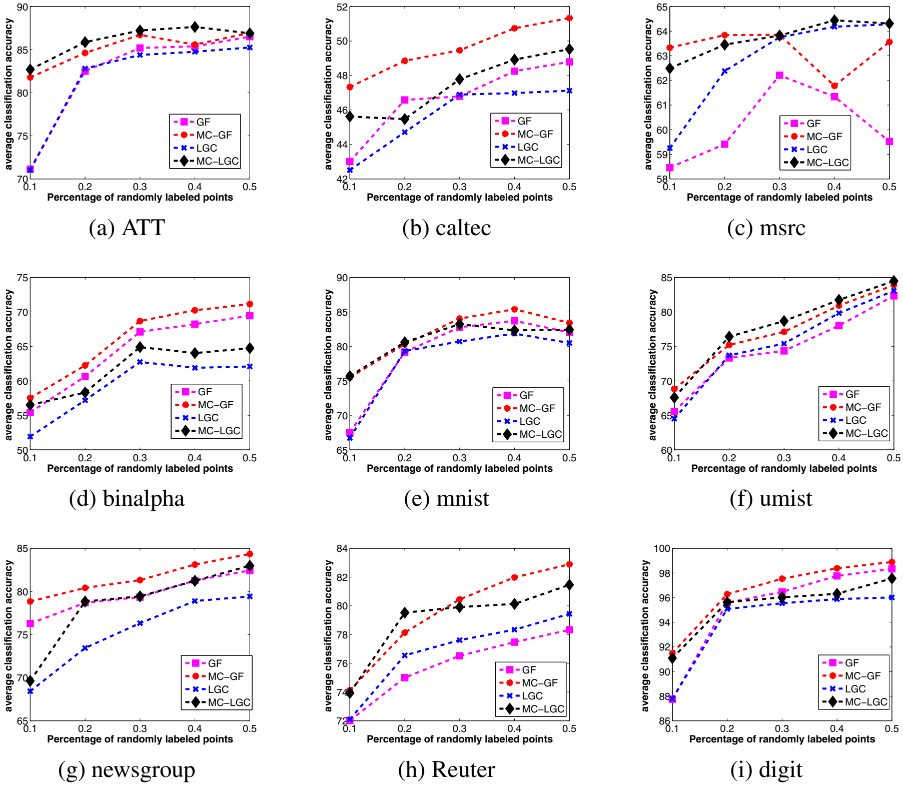
(a) ATT (b) caltec (c) msrc

(d) binalpha (e) mnist (f) umist

(g) newsgroup (h) Reuter (i) digit

Fig. 3. Experiments results on 4 methods of Generalized Preferential Random Walks: GF, method1, method2(=LGC), method3. x-axis represents the different α settings($\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$), y-axis is the average classification accuracy over 10 independent runs.

method 1. In all the methods except in GF, parameter α will influence the semi-supervised classification results. For image datasets, we use Gaussian kernel to construct the graph edge weights $\mathbf{W}_{ij} = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}$, where γ is fine tuned according to [10]. For text datasets, we use linear kernel to compute graph similarity. We randomly select 20% of all data as the training data. In Fig.3, we show the average classification



(a) ATT (b) caltec (c) msrc

(d) binalpha (e) mnist (f) umist

(g) newsgroup (h) Reuter (i) digit

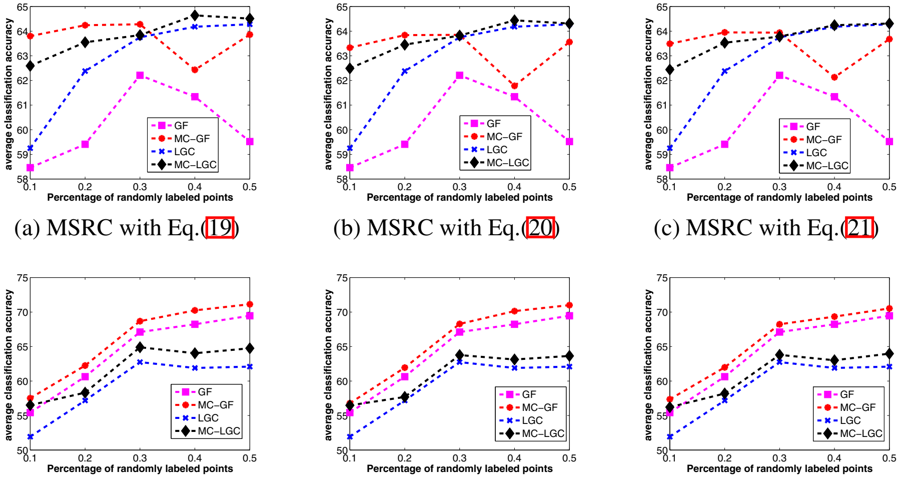
Fig. 4. Experiments results on 4 methods of label propagation: GF, MC-GF, LGC, MC-LGC. x-axis represents the different percentage of labeled data, y-axis is the average classification accuracy over 10 independent runs

results on 4 methods (GF, method1, method2(=LGC), method3) by using 5-fold cross-validation. In Fig.3, x-axis represents different α settings($\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$), y-axis is the average classification accuracy over 10 independent runs.

Experiment Result Analysis. From Fig. 3, we can observe: (1) method 1 and GF perform well on all the datasets; (2) parameter α does not influence very much for the classification results obtained from method 1; (3) method 2 and 3 perform reasonably well when $\alpha \leq 0.5$, but their performances degrade much when α is approaching 1.

7.3 Experiment Results on Maximum Consistency Algorithm

We compare maximum consistency algorithm with standard LGC and GF methods. The α in LGC and MC-LGC methods are set to $\alpha = 0.5$ as suggested in [13]. We use Eq.(19) as the discriminant score in the balanced class expansion process. The maximum iteration time T is set according to the number of data points in the unlabeled



(a) MSRC with Eq.(19)

(b) MSRC with Eq.(20)

(c) MSRC with Eq.(21)

(d) binalpha with Eq.(19)

(e) binalpha with Eq.(20)

(f) binalpha with Eq.(21)

Fig. 5. Experiments results on 4 methods of label propagation: GF, MC-GF, LGC, MC-LGC using different discriminant score computations of Eqs.(19,20 and 21) on datasets MSRC and binalpha. x-axis represents the different percentage of labeled data, y-axis is the average classification accuracy over 10 independent runs

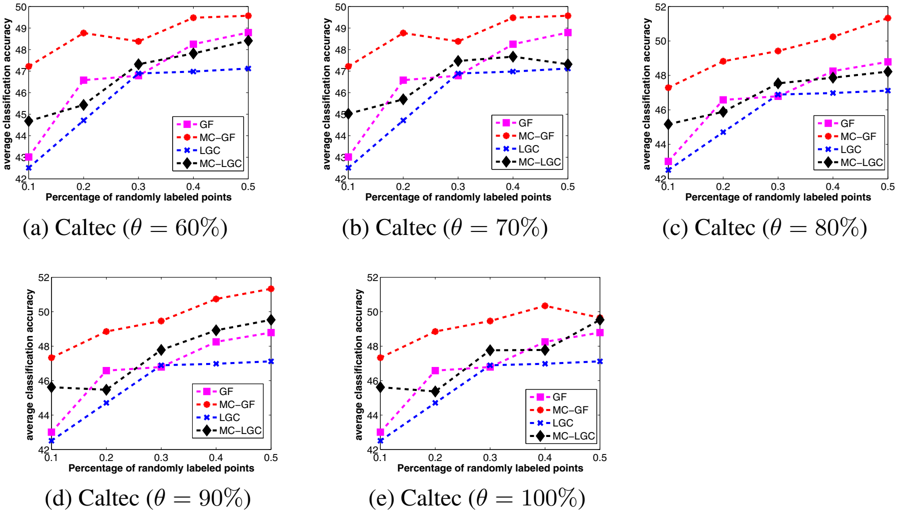
pool. If there are more than $\theta = 90\%$ of the whole data labeled, we stop the proposed maximum consistency algorithm, and do one-shot label propagation.

We show the classification results of 4 methods (LGC, MC-LGC, GF, MC-GF) by randomly selecting different percentages of labeled data in Fig.4, where x-axis represents different percentages of labeled data (i.e., $10\%, 20\%, \cdots$), and y-axis is the average classification accuracy over 10 independent runs.

Experiment Results Analysis. From Fig. 4, we observe, (1) MC-LGC consistently performs better than LGC especially when the percentage of labeled data is very small (e.g., 10%); (2) MC-GF performs much better than GF; (3) on text dataset, MC-GF's superiority is much more significant (more than 5% improvement). Next, we discuss maximum consistency algorithm experiment results with different parameter settings.

7.4 Discussion on the Parameter Settings of Maximum Consistency Algorithm

Discussion on Discriminant Score Computation. Discriminant score computation is very important for the decision of data to be propagated. The first issue is how to compute the discriminant score. Here we show the experiment results of classification when alternative discriminant score computation formulations of Eq.(20, 21) are used. The other settings of the experiments are the same as those described in §7.3. Fig. 5 shows the classification results of 4 methods of label propagation (GF, MC-GF, LGC, MC-LGC) by using different discriminant score computations of Eqs.(19, 20, 21) on datasets MSRC and binalpha. We observe that, most of the time, the classification results obtained from Eq.(19) are slightly better on both datasets for both MC-GF and MC-LGC methods. These experiment results suggest us to use Eq.(19) in our algorithm.



(a) Caltec ($\theta = 60\%$)

(b) Caltec ($\theta = 70\%$)

(c) Caltec ($\theta = 80\%$)

(d) Caltec ($\theta = 90\%$)

(e) Caltec ($\theta = 100\%$)

Fig. 6. Experiments results on 4 methods of label propagation: GF, MC-GF, LGC, MC-LGC by using different parameter θ on dataset Caltec. x-axis represents the different percentage of labeled data, y-axis is the average classification accuracy over 10 independent runs

Discussion on the Iteration Number. Another key parameter is related to the extent to which the procedure is designed for maximizing the label assignment consistency. As described in §7.3, we use the number of labeled data points in labeled pool as a criteria to stop our algorithm. We use parameter θ to represent the percentage of *currently* labeled data of the whole dataset. In §7.3, we set $\theta = 0.9$. We try different settings of $\theta = \{60\%, 70\%, 80\%, 90\%, 100\%\}$ and report the experiment results on dataset Caltec in Fig. 6. The other settings of the experiments are the same as those described in §7.3. We find, on most of the datasets, if we set $\theta = 90\%$, we can achieve the best results. Thus we set $\theta = 90\%$ as the default setting for our maximum consistency algorithm.

8 Conclusion

We analyze the relations between 3 methods of generalized preferential random walks and label propagation methods. A maximum consistency algorithm is presented to improve current label propagation methods. Extensive experiments on 9 datasets show the effectiveness of MC algorithm and different generalized preferential random walks.

Acknowledgments. This work is supported partially by NSF-CCF-0939187, NSF-CCF-0917274, NSF-DMS-0915228.

References

1. Pearson, K.: The problem of the Random Walk. Nature (1905)
2. Craswell, N., Szummer, M.: Random walks on the click graph. In: SIGIR (2007)
3. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: WSDM (2011)

4. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University Database Group (1998)
5. Glen, J., Jennifer, W.: Scaling personalized web search. Technical report, Stanford University Database Group (2002)
6. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
7. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin-Madison (2008)
8. Blum, A., Mitchell, T.M.: Combining labeled and unlabeled sata with co-training. In: COLT, pp. 92–100 (1998)
9. Joachims, T.: Transductive learning via spectral graph partitioning. In: ICML, pp. 290–297 (2003)
10. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 20th International Conference on Machine Learning, pp. 912–919 (2003)
11. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Scholkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems, vol. 16, pp. 321–328 (2004)
12. Ding, C.H.Q., Jin, R., Li, T., Simon, H.D.: A learning framework using green's function and kernel regularization with application to recommender system. In: KDD, pp. 260–269 (2007)
13. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems 16, pp. 321–328. MIT Press (2004)
14. Minkov, E., Cohen, W.W.: Learning to rank typed graph walks: Local and global approaches. In: WebKDD and SNA-KDD Joint Workshop (2007)
15. Tong, H., Faloutsos, C., Pan, J.-Y.: Fast Random Walk with Restart and Its Applications. In: ICDM (2006)
16. Tong, H., Faloutsos, C.: Center-Piece Subgraphs: Problem Definition and Fast solutions. In: KDD (2006)
17. Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: KDD (2008)
18. Koutra, D., Ke, T.-Y., Kang, U., Horng (Polo) Chau, D., Pao, H.-K.K., Faloutsos, C.: Unifying Guilt-by-Association Approaches: Theorems and Fast Algorithms. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 245–260. Springer, Heidelberg (2011)
19. Pearl, J.: Reverend bayes on inference engines: A distributed hierarchical approach. In: AAAI, pp. 133–136 (1982)
20. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29, 93–106 (2008)
21. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory 51, 2282–2312 (2005)
22. Flake, G.W., Lawrence, S., Giles, C.L., Coetzee, F.M.: Self-Organization and Identification of Web Communities. IEEE Computer 35 (2002)
23. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 2278–2324 (1998)
24. Dueck, D., Frey, B.J.: Non-metric affinity propagation for unsupervised image categorization. In: ICCV (2007)
25. Lee, Y.J., Grauman, K.: Foreground focus: Unsupervised learning from partially matching images. International Journal of Computer Vision 85, 143–166 (2009)

Semi-supervised Multi-label Classification
A Simultaneous Large-Margin, Subspace Learning Approach

Yuhong Guo and Dale Schuurmans

[1] Department of Computer and Information Sciences
Temple University
Philadelphia, PA 19122, USA
[2] Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada

Abstract. Labeled data is often sparse in common learning scenarios, either because it is too time consuming or too expensive to obtain, while unlabeled data is almost always plentiful. This asymmetry is exacerbated in *multi-label* learning, where the labeling process is more complex than in the single label case. Although it is important to consider *semi-supervised* methods for multi-label learning, as it is in other learning scenarios, surprisingly, few proposals have been investigated for this particular problem. In this paper, we present a new semi-supervised multi-label learning method that combines large-margin multi-label classification with unsupervised subspace learning. We propose an algorithm that learns a subspace representation of the labeled and unlabeled inputs, while simultaneously training a supervised large-margin multi-label classifier on the labeled portion. Although joint training of these two interacting components might appear intractable, we exploit recent developments in induced matrix norm optimization to show that these two problems can be solved jointly, globally and efficiently. In particular, we develop an efficient training procedure based on subgradient search and a simple coordinate descent strategy. An experimental evaluation demonstrates that semi-supervised subspace learning can improve the performance of corresponding supervised multi-label learning methods.

Keywords: semi-supervised multi-label learning, subspace learning.

1 Introduction

In many real world data analysis problems, complex data objects such as documents, webpages, images and videos can be simultaneously assigned into multiple categories, and hence have multiple class labels. *Multi-label* classification is an important supervised learning problem that has received significant attention in the machine learning research literature. Although the earliest work on this problem simply reduced multi-label classification to a set of independent binary classification problems [1], it was quickly realized that such an approach

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 355–370, 2012.

was unsatisfactory [2] since such labels are usually not independent: most often they exhibit strong correlations. Capturing label correlations in an effective, yet tractable manner has led to a diverse set of formulations for multi-label learning, including pairwise label dependency methods [3, 4], large-margin methods [5–8], ranking based large margin methods [9–12], structure exploitation methods [13–18] and others. Of these, it has recently been observed that *large-margin* based approaches offer effective and efficient multi-label learning methods. In particular, it has been shown that a large-margin formulation based on minimizing a "calibrated separation ranking loss" demonstrates state-of-the-art performance in multi-label text categorization [8].

However, just as in any supervised learning scenario, a key bottleneck is obtaining sufficient labeled data to achieve reasonable generalization performance. In practice, one often encounters a significant amount of unlabeled data, even while labeled examples remain scarce, since labeling is an expensive and time-consuming process. *This issue is even more salient in multi-label learning*, since manually assigning multiple labels, correctly, is more challenging than assigning atomic labels. Thus, we address the challenge of exploiting significant unlabeled data to reduce the amount of labeled training data required for effective multi-label classification. Although *supervised* multi-label learning has received significant attention, *semi-supervised* multi-label learning is far from being well explored. A handful of preliminary studies have explored semi-supervised multi-label learning, using approaches such as non-negative matrix factorization [19], graph-based methods [20], and dimensionality reduction [21]. Unfortunately, these proposals rely on local optimization schemes for training, and do not offer reliable off-the-shelf procedures that protect end-users from local minima.

In this work, we propose a new approach for exploiting unlabeled data to help multi-label learning in a transductive setting, by simultaneously learning the underlying subspace feature representations of the data with a large margin multi-label classification model. Automatically discovering useful feature representations of data has been a long standing research of machine learning—from early unsupervised approaches, such as principal component analysis (PCA)—to recent supervised convex feature learning, such as multi-task feature learning [22]. Here we exploit recent results for semi-supervised convex subspace learning, which we adapt to large-margin multi-label classification. Our approach is based on two key recent ideas: (1) using calibrated separation ranking loss for large margin multi-label classification [8], and (2) using induced matrix norms to efficiently combine subspace learning with semi-supervised training [23]. By introducing a structured regularizer on the learned representation, and exploiting a particular induced matrix norm, we formulate the semi-supervised multi-label learning problem as a convex max-min optimization problem with no local maxima or minima. We then develop a specialized subgradient coordinate descent algorithm to solve the training problem efficiently, recovering a global solution.

The goal is to discover a subspace feature representation that captures discriminative structure that is not only shared across labeled and unlabeled data, but also shared across the multiple labels. Our experimental results demonstrate that the proposed method can surpass the performance of some state-of-the-art supervised results for multi-label text categorization.

The remainder of the paper is organized as follows. After first introducing basic background concepts and notation, we review previous work on large margin multi-label classification in Section 2, with a particular emphasis on the calibrated separation ranking loss used for state-of-the-art multi-label text categorization [8]. Based on this particular multi-label classification approach, we then present a semi-supervised formulation in Section 3 that exploits implicit subspace learning through structured matrix norm regularization. An efficient global optimization algorithm is then presented in Section 4. Finally, we present an experimental evaluation in Section 5 and conclude the paper with a discussion of future research directions in Section 6.

1.1 Preliminaries: Definitions and Notation

Throughout this paper we will use capital letters to denote matrices, bold non-capital letters to denote column vectors, and regular non-capital letters to denote scalars, unless special declaration is given.

We use I_d to denote a $d \times d$ identity matrix; and use $\mathbf{1}$ to denote a column vector with all 1 entries, generally assuming its length can be inferred from context. Given a vector $\mathbf{x}$, $\|\mathbf{x}\|_2$ denotes its Euclidean norm.

Given a matrix X, $\|X\|_F^2$ denotes its Frobenius norm; the block norm $\|X\|_{p,1}$ is defined as $\|X\|_{p,1} = (\sum_i (\sum_j |X_{ij}|^p)^{\frac{1}{p}})$; and the trace norm is defined as $\|X\|_{tr} = \sum_i \sigma_i(X)$, where $\sigma_i(X)$ denotes the ith singular value of X. We use $X_{i:}$ to denote the ith row of a matrix X, use $X_{:j}$ to denote the jth column of X, and use X_{ij} to denote the entry at the ith row and jth column of X. We also need to make use of a general form of induced matrix norm given by the definition $\|X\|_{(\mathcal{Z},p)} := \max_{\mathbf{z}\in\mathcal{Z}} \|X\mathbf{z}\|_p$. It can be shown [23] that this defines a valid matrix norm for any bounded closed set $\mathcal{Z} \subset \mathbb{R}^n$ such that $\text{span}(\mathcal{Z}) = \mathbb{R}^n$ and any $1 \leq p \leq \infty$. Finally, for matrices, we use $\|X\|$ to refer to a generic norm on X, and $\|Y\|^*$ to denote its conjugate norm. The conjugate satisfies $\|Y\|^* = \max_{\|X\|\leq 1} \text{tr}(X^\top Y)$ and $\|X\|^{**} = \|X\|$, where tr denotes trace.

2 Background

Our main formulation is based on combining two key components: an effective large-margin formulation for multi-label learning, and an efficient approach for automated representation learning that avoids local optima.

2.1 Large Margin Multi-label Classification

Multi-label classification is a widely studied problem in supervised machine learning, for which large margin methods provide one of the state-of-the-art approaches. By maximizing discriminative classification margins, expressed by different loss functions, supervised large margin learning methods are both efficient, and demonstrate good generalization performance.

In particular, [8] has recently proposed an effective method for supervised multi-label learning that exploits the dependence structure between labels in a simple yet effective manner. The basic idea is to simultaneously train a set of L predictors, one for each class label, but under a coordinated loss: the calibrated separation ranking loss captures the sum of two hinge losses, one of which is between the prediction value of the least positive labeled class and the prediction value of a threshold dummy class, and the other of which is between the prediction value of the least negative labeled class and the prediction value of the threshold dummy class.

More formally, in the supervised multi-label learning setting, one is given an input data matrix $X \in \mathbb{R}^{t\times d}$ and label indicator matrix $Y \in \{0,1\}^{t\times L}$, where L denotes the number of classes. We also assume a feature mapping function $\phi(\cdot)$ is fixed. Then, given an input instance $\mathbf{x}$, the L dimensional response vector $\mathbf{s}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top W$ is recovered using W, giving a "score" for each label. These scores will be compared to a threshold to determine which labels are to be predicted. Then the calibrated separation ranking loss is given by

$$\max_{l\in Y_{i:}}(1+s_0(X_{i:})-s_l(X_{i:}))_+ + \max_{\bar{l}\in \bar{Y}_{i:}}(1+s_{\bar{l}}(X_{i:})-s_0(X_{i:}))_+. \tag{1}$$

So, for example, given a test example $\mathbf{x}$, its classification is determined by $y_l^* = \arg\max_{y_l\in\{0,1\}} y_l(s_l(\mathbf{x})-s_0(\mathbf{x}))$.

It is shown in [8] that minimizing this loss under standard squared regularization can be formulated as a standard convex quadratic minimization problem

$$\begin{aligned} \min_{W,\mathbf{u},\boldsymbol{\xi},\boldsymbol{\eta}} \quad & \frac{\alpha}{2}(\|W\|_F^2+\|\mathbf{u}\|_2^2)+\mathbf{1}^\top\boldsymbol{\xi}+\mathbf{1}^\top\boldsymbol{\eta} \\ \text{subject to} \quad & \boldsymbol{\xi}_i \geq 1+X_{i:}(\mathbf{u}-W_{:l}) \ \text{ for } l\in Y_{i:}, \forall i=1\cdots t \\ & \boldsymbol{\eta}_i \geq 1-X_{i:}(\mathbf{u}-W_{:\bar{l}}) \ \text{ for } \bar{l}\in \bar{Y}_{i:}, \forall i=1\cdots t \\ & \boldsymbol{\xi}\geq 0, \boldsymbol{\eta}\geq 0 \end{aligned} \tag{2}$$

where $l \in Y_{i:}$ lists through the indices of all entries of $Y_{i:}$ that contain 1 values, and $\bar{Y}$ denotes the complementary of Y, i.e., $\bar{Y} = 1 - Y$. Obviously the loss function only captures the classification relevant separation ranking that separate positive labels from negative labels for each instance, instead of the pairwise rankings among all label pairs in [9]. By conducting calibrated separation ranking, the label separation on new testing instances can be automatically determined using the trained predictors.

In [8] this approach is shown to demonstrate superior generalization and efficiency in supervised multi-label text categorization, so we make use of this loss in our semi-supervised formulation.

2.2 Unsupervised Representation Learning

To allow unlabeled data to influence the training of a multi-label classifier we consider the approach of learning a new input data representation that makes the label correlations more apparent. We begin by adopting a recent approach to representation learning that offers a tractable way to learn a latent representation and a data reconstruction model.

Initially, consider the case where one is just given unlabeled data X. A simple goal for representation learning is to learn an $m \times d$ dictionary B of m basis vectors, and a $t \times m$ representation matrix $\varPsi$ containing new feature vectors of length m, so that X can be accurately reconstructed from $\hat{X} = \varPsi B$. To measure approximation error we consider the loss function $\frac{1}{2}\|\hat{X} - X\|_F^2$. Note that the factorization $\hat{X} = \varPsi B$ is invariant to reciprocal rescalings of B and $\varPhi$, so to avoid degeneracy their individual magnitudes have to be controlled. We will assume that each row $B_{i:}$ of B is constrained to belong to a bounded closed convex set $\mathcal{B} = \{\mathbf{b} : \|\mathbf{b}\|_2 \leq 1\}$. The generic training problem can be expressed

$$\min_{B \in \mathcal{B}^m} \min_{\varPsi} \frac{1}{2}\|\varPsi B - X\|_F^2 + \gamma \|\varPsi^\top\|_{p,1} \tag{3}$$

where $\gamma \geq 0$ is a trade-off parameter. Some standard approaches to representation learning can be recovered by particular choices of p in (3). For example, a standard form of *sparse coding* can be recovered by choosing $p = 1$ [24]. Instead, choosing $p = 2$ results in a regularizer that encourages entire columns $\varPsi_{:j}$ (features) to become sparse [25] while otherwise only smoothing the rows, hence implicitly reducing the dimensionality of the learned representation $\varPsi$.

Unfortunately, the straightforward formulation (3) is not jointly convex in B and $\varPsi$, and even recent formulations resort to local minimization strategies. However, a key observation is that the training problem can be solved globally if the number of learned features m is indirectly controlled through the use of the $\|\varPsi^\top\|_{p,1}$ regularizer. As noted, for $p > 1$, such a regularizer will already naturally encourage entire columns $\varPsi_{:j}$ (features) to become sparse [25]. A key result that leads to a tractable reformulation is the following identity from [23, 26].

Proposition 1. *[23, Theorem 1]:*

$$\min_{B \in \mathcal{B}^\infty} \min_{\varPsi} \frac{1}{2}\|\varPsi B - X\|_F^2 + \gamma \|\varPsi^\top\|_{p,1} \quad = \quad \min_{\hat{X}} \frac{1}{2}\|\hat{X} - X\|_F^2 + \gamma \|\hat{X}\|^*_{(\mathcal{B},p^*)} \tag{4}$$

where $\min_{B \in \mathcal{B}^\infty}$ *denotes* $\min_{m \in \mathbb{N}} \min_{B \in \mathcal{B}^m}$*, and* $\|\cdot\|^*_{(\mathcal{B},p^*)}$ *is the conjugate of an induced matrix norm.*

The latter problem is convex, and for $p = 1$ or $p = 2$ can be readily solved for $\hat{X}$, after which the optimal factors B and $\varPsi$ can be readily recovered [23, 26].

3 Simultaneous Multi-label Classification and Representation Learning

Our main contribution in this paper is to combine these two components to formulate a multi-label classification and representation learning framework that uses unlabeled data to guide the learning of a multi-label classifier. Such an approach enables semi-supervised learning, where unlabeled data assists in the otherwise supervised training of a classification model. We will investigate semi-supervised learning in a transductive setting, where a set of labeled and unlabeled data is provided, and one seeks an accurate labeling over the unlabeled portion.[1] The main advantage of the proposed formulation is that it admits an efficient global training scheme that combines multi-label classification with representation learning.

Note that global training schemes offer a significant advantage to the end-user, since they do not need to concern themselves with the inner workings of any particular solver. Rather, it is sufficient to focus on understanding the nature of the problem formulation, and the solver can be used as a black box. This separation of implementation from specification frees the end-user to focus on engineering useful features, or imposing trade-offs between training errors and regularization penalties, without having to understand the inner workings of a solver. In this paper, however, we need to show that a suitably efficient solver can exist, which we do in the next section.

To develop a combined formulation of multi-label classification and representation learning, consider the following set-up. Let $X \in \mathbb{R}^{t \times d}$ be the input feature matrix and let $X^\ell \in \mathbb{R}^{t_\ell \times d}$ be the labeled submatrix formed by the first t_ℓ rows of X, where $t_\ell + t_u = t$. Let $Y \in \{0,1\}^{t_\ell \times L}$ be the label matrix over the supervised portion.

We would like to learn a $(t_l + t_u) \times m$ representation matrix $\Psi = [\Psi_l; \Psi_u]$, an $m \times d$ basis dictionary B, and an $m \times L$ prediction model W, such that $X = [X_l; X_u]$ can be reconstructed from $\hat{X} = \Psi B$, and Y can be reconstructed from $\hat{Y} = \Psi_l W$. To accommodate the offset in the calibrated separation ranking loss (1) we consider a linear prediction function over the subspace representation $\Psi(W_{:l} - \mathbf{u})$. Then by combining (2) and (3) we reach the joint training formulation

$$
\begin{aligned}
\min_{\Psi, B \in \mathcal{B}^m} \; \min_{W, \mathbf{u}, \boldsymbol{\xi}, \boldsymbol{\eta}} \;\; & \frac{\alpha}{2}(\|W\|_F^2 + \|\mathbf{u}\|_2^2) + \mathbf{1}^\top \boldsymbol{\xi} + \mathbf{1}^\top \boldsymbol{\eta} + \frac{\beta}{2}\|X - \Psi B\|_F^2 + \gamma \|\Psi^\top\|_{p,1} \quad (5) \\
\text{subject to} \;\; & \boldsymbol{\xi}_i \ge 1 + \Psi_{i:}(\mathbf{u} - W_{:l}) \;\; \text{for } l \in Y_{i:}, \forall i = 1 \cdots t_\ell \\
& \boldsymbol{\eta}_i \ge 1 - \Psi_{i:}(\mathbf{u} - W_{:\bar{l}}) \;\; \text{for } \bar{l} \in \bar{Y}_{i:}, \forall i = 1 \cdots t_\ell \\
& \boldsymbol{\xi} \ge 0, \boldsymbol{\eta} \ge 0
\end{aligned}
$$

where now the multi-label predictions are made from the learned representation Ψ, which is the only component that connects the multi-label training problem to the representation learning problem. Note that for $p > 1$ the regularizer $\|\Psi^\top\|_{p,1}$ will tend to reduce the dimensionality of the learned representation Ψ, which gives an automated form of subspace learning directly coupled to the multi-label training problem. Unfortunately, (5) does not immediately offer a plausible global training algorithm that avoids local minima: although it is straightforward

[1] The approach we propose here is in principle extendible to a semi-supervised learning scenario where the test data is not available during training. However, such out-of-sample classification entails significant additional technicality that is currently left to future work.

to observe that (5) is convex in each of the components B, W, Ψ, $\mathbf{u}$, $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ given the others, it is not jointly convex. Fortunately Proposition 1 can be generalized to accommodate the more general formulation given here, as we now show.

3.1 Equivalent Reformulation as a Convex Problem

Unlike staged training procedures that separate the unsupervised from the supervised phase [27], and previous work on semi-supervised dimensionality reduction that relies on alternating minimization [28], here we demonstrate a jointly convex formulation that allows all components to be trained simultaneously.

Let $M = \Psi B$ and $Z = \Psi(W - [\mathbf{u}, \cdots, \mathbf{u}])$ denote the reconstruction and response matrices respectively. To simplify the development below, we set $\mathbf{u} = 0$ and therefore let $Z = \Psi W$ and $M = \Psi B$; hence $Z \in \mathbb{R}^{t \times L}$ and $M \in \mathbb{R}^{t \times d}$. Substituting this into (5) yields

$$\begin{aligned} \min_{\Psi, B \in \mathcal{B}^m, W, \boldsymbol{\xi}, \boldsymbol{\eta}} \quad \min_{Z=\Psi W, M=\Psi B} \quad & \frac{\alpha}{2}\|W\|_F^2 + \mathbf{1}^\top \boldsymbol{\xi} + \mathbf{1}^\top \boldsymbol{\eta} + \frac{\beta}{2}\|X - M\|_F^2 + \gamma \|\Psi^\top\|_{p,1} \\ \text{subject to} \quad & \boldsymbol{\xi}_i \geq 1 - Z_{il} \ \text{ for } l \in Y_{i:}, \forall i = 1 \cdots t_\ell \\ & \boldsymbol{\eta}_i \geq 1 + Z_{i\bar{l}} \ \text{ for } \bar{l} \in \bar{Y}_{i:}, \forall i = 1 \cdots t_\ell \\ & \boldsymbol{\xi} \geq 0, \boldsymbol{\eta} \geq 0. \end{aligned} \tag{6}$$

To achieve compatibility with the reformulation exploited by Proposition 1 we replace the regularization penalty $\|W\|_F^2$ with a constraint on the norms of rows $W_{i:}$ in W. In particular, we constrain each $W_{i:}$ to the bounded closed set $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq \alpha\}$. This leads to a slightly modified formulation

$$\begin{aligned} \min_{\Psi, B \in \mathcal{B}^m, W \in \mathcal{W}^m, \boldsymbol{\xi}, \boldsymbol{\eta}} \quad \min_{Z=\Psi W, M=\Psi B} \quad & \mathbf{1}^\top \boldsymbol{\xi} + \mathbf{1}^\top \boldsymbol{\eta} + \frac{\beta}{2}\|X - M\|_F^2 + \gamma \|\Psi^\top\|_{p,1} \\ \text{subject to} \quad & \boldsymbol{\xi}_i \geq 1 - Z_{il} \ \text{ for } l \in Y_{i:}, \forall i = 1 \cdots t_\ell \\ & \boldsymbol{\eta}_i \geq 1 + Z_{i\bar{l}} \ \text{ for } \bar{l} \in \bar{Y}_{i:}, \forall i = 1 \cdots t_\ell \\ & \boldsymbol{\xi} \geq 0, \boldsymbol{\eta} \geq 0. \end{aligned} \tag{7}$$

Finally, by relaxing the feature number m and instead allowing the rank reducing regularizer $\|\Psi^\top\|_{p,1}$ to automatically choose the dimension, [23, Proposition 3] shows that the problem (7) is equivalent to

$$\begin{aligned} \min_{Z, M, \boldsymbol{\xi}, \boldsymbol{\eta}} \quad & \mathbf{1}^\top \boldsymbol{\xi} + \mathbf{1}^\top \boldsymbol{\eta} + \frac{\beta}{2}\|X - M\|_F^2 + \gamma \|[M, Z]\|_{(\mathcal{U}, p^*)}^* \\ \text{subject to} \quad & \boldsymbol{\xi}_i \geq 1 - Z_{il} \ \text{ for } l \in Y_{i:}, \forall i = 1 \cdots t_\ell \\ & \boldsymbol{\eta}_i \geq 1 + Z_{i\bar{l}} \ \text{ for } \bar{l} \in \bar{Y}_{i:}, \forall i = 1 \cdots t_\ell \\ & \boldsymbol{\xi} \geq 0, \boldsymbol{\eta} \geq 0 \end{aligned} \tag{8}$$

where as in Proposition 1, $\|\cdot\|_{(\mathcal{U}, p^*)}$ is a conjugate of an induced matrix norm, but now with respect to the closed bounded set $\mathcal{U} := \{[\mathbf{b}; \mathbf{w}] : \|\mathbf{b}\|_2 \leq 1 \text{ and } \|\mathbf{w}\|_2 \leq \alpha\}$.

Importantly, the problem (8) is jointly convex in Z, M, $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$, since norms are always convex. For $p = 1$ or $p = 2$, given an optimal pair $[M, Z]$, the underlying factors Ψ, B and W can be efficiently recovered using a procedure outlined in [23]. However, for the purposes of transduction, Z itself is sufficient for determining the label predictions on the unlabeled data, so this extra recovery procedure can be bypassed.

3.2 Tractable Special Case

Even though the problem (8) is convex, it is not guaranteed to be tractable, since not every induced matrix norm is tractable to compute [29]. However, the important special cases of $p = 1$ and $p = 2$ both allow the induced norm $\|\cdot\|_{(\mathcal{U},p^*)}$ to be efficiently evaluated. In particular, for $p = 2$, [23] establishes the following useful characterization.

Proposition 2. *[23, Lemma 5]:* $\|[M, Z]\|^*_{(\mathcal{U},2)} = \max_{\rho\geq 0} \|D_\rho^{-1}[M, Z]^\top\|_{tr}$ *where*

$$D_\rho = \begin{bmatrix} \sqrt{1+\alpha^2\rho}\, I_d & 0 \\ 0 & \sqrt{\alpha^2 + \frac{1}{\rho}}\, I_L \end{bmatrix}. \tag{9}$$

This proposition shows that for the case $p = 2$ the conjugate induced norm can be efficiently computed: all that is required is a line search over a scalar variable $\rho \geq 0$, where for each value of ρ the inner calculation can be efficiently evaluated by computing the singular value decomposition of $[M, Z]D_\rho^{-1}$.

Below we find it more convenient to work with a re-parameterized version of the calculation.

Proposition 3. $\max\limits_{\rho\geq 0} \|D_\rho^{-1}[M, Z]^\top\|_{tr} = \max\limits_{0\leq\theta\leq 1} \|E_\theta[M, Z]^\top\|_{tr}$ *where*

$$E_\theta = \begin{bmatrix} \sqrt{\theta}\, I_d & 0 \\ 0 & \frac{\sqrt{1-\theta}}{\alpha}\, I_L \end{bmatrix}. \tag{10}$$

This proposition is easy to establish by noting the relationships $D_\rho^{-1} = E_{\frac{1}{\alpha^2\rho+1}}$, $E_\theta^{-1} = D_{\frac{1-\theta}{\alpha^2\theta}}$, $\theta = \frac{1}{\alpha^2\rho+1}$, and $\rho = \frac{1-\theta}{\alpha^2\theta}$, hence optimizing with D_ρ^{-1} over the range $\rho \geq 0$ is equivalent to optimizing with E_θ over the range $0 \leq \theta \leq 1$.

Thus, we obtain the following convex optimization problem that is equivalent to (8) for the special case when $p = 2$:

$$\begin{aligned} \max_{0\leq\theta\leq 1}\ \min_{Z,M,\boldsymbol{\xi},\boldsymbol{\eta}}\ & \mathbf{1}^\top\boldsymbol{\xi} + \mathbf{1}^\top\boldsymbol{\eta} + \frac{\beta}{2}\|X - M\|_F^2 + \gamma\|E_\theta[M, Z]^\top\|_{tr} \\ \text{subject to }\ & \boldsymbol{\xi}_i \geq 1 - Z_{il} \text{ for } l \in Y_{i:}, \forall i = 1\cdots t_\ell \\ & \boldsymbol{\eta}_i \geq 1 - Z_{il} \text{ for } \bar{l} \in \bar{Y}_{i:}, \forall i = 1\cdots t_\ell \\ & \boldsymbol{\xi} \geq 0, \boldsymbol{\eta} \geq 0. \end{aligned} \tag{11}$$

To verify that this problem has no local optima, first note that the inner problem is convex in Z, M, $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ for each fixed θ. Furthermore, it can be shown that

$\|E_\theta[M,Z]^\top\|_{tr}$ is concave in θ. Since a pointwise minimum of concave functions is concave, the outer optimization in θ also has no local maxima. To solve (11), all one has to do is run a simple outer concave maximization over a scalar variable, while each inner minimization is a standard convex minimization. Essentially, the inner problem has the same complexity as the standard multi-label learning problem, which only has to be repeated a few times (say, around 10) to achieve an accurate solution.

4 Optimization Algorithm

The semi-supervised optimization problem we formulated above in (11) is a convex optimization problem but with a non-smooth trace norm. To develop an efficient optimization algorithm for it, we first derive an equivalent reformulation following a well-known variational formulation of the trace norm [22, 30]:

Proposition 4. *Let $Q \in \mathbb{R}^{t\times d}$. The trace norm of Q is equal to*

$$\|Q\|_{tr} = \frac{1}{2}\inf_{S\succeq 0} tr(Q^\top S^{-1}Q) + tr(S), \tag{12}$$

and the infimum is achieved for $S = (QQ^\top)^{1/2}$.

Following this proposition, we can reformulate (11) as the following

$$\begin{aligned} \max_{0\le\theta\le 1}\ \min_{Z,M,\boldsymbol{\xi},\boldsymbol{\eta}}\ \inf_{S\succeq 0}\ & \mathbf{1}^\top\boldsymbol{\xi} + \mathbf{1}^\top\boldsymbol{\eta} + \frac{\beta}{2}\|X-M\|_F^2 \\ & + \frac{\gamma}{2}\mathrm{tr}([M,Z]E_\theta S^{-1}E_\theta[M,Z]^\top) + \frac{\gamma}{2}\mathrm{tr}(S) \\ \text{subject to}\ \ & \boldsymbol{\xi}_i \ge 1 - Z_{il}\ \text{ for } l \in Y_{i:}, \forall i = 1\cdots t_\ell \\ & \boldsymbol{\eta}_i \ge 1 + Z_{i\bar{l}}\ \text{ for } \bar{l} \in \bar{Y}_{i:}, \forall i = 1\cdots t_\ell \\ & \boldsymbol{\xi} \ge 0, \boldsymbol{\eta} \ge 0 \end{aligned} \tag{13}$$

which maintains the convexity of the original formulation of (11). Although the reformulated problem remains a non-smooth max-min optimization problem, an efficient optimization procedure can still be developed. In particular, we develop a simple subgradient-based binary line search procedure, combined with a block-descent inner minimization, to solve this problem.

First, consider the max-min optimization problem in (13) as a non-smooth concave optimization problem over θ

$$\max_{0\le\theta\le 1}\quad f(\theta) \tag{14}$$

where the objective function is a non-smooth function defined by a convex minimization problem

$$\begin{aligned} f(\theta) = \min_{M,\,\{Z,\boldsymbol{\xi},\boldsymbol{\eta}\}\in\mathcal{C}}\ \inf_{S\succeq 0}\ & \mathbf{1}^\top\boldsymbol{\xi} + \mathbf{1}^\top\boldsymbol{\eta} + \frac{\beta}{2}\|X-M\|_F^2 \\ & + \frac{\gamma}{2}\mathrm{tr}([M,Z]E_\theta S^{-1}E_\theta[M,Z]^\top) + \frac{\gamma}{2}\mathrm{tr}(S). \end{aligned} \tag{15}$$

Here we use $\mathcal{C}$ to denote the feasible region defined by the linear constraints in (13) over variables $\{Z, \boldsymbol{\xi}, \boldsymbol{\eta}\}$. For such a non-smooth optimization problem, subgradient-based methods such as proximal bundle methods can be generally applied. Nevertheless, we develop a much simpler *binary line search* procedure to tackle this specific one dimensional optimization problem over θ.

4.1 Binary Line Search

The idea of binary line search is to iteratively reduce the searching region (interval) for the optimal θ value, eliminating at least half of the feasible region each time. At the beginning of the binary line search, θ is upper-bounded by $V_u = 1$ and lower-bounded by $V_\ell = 0$, which is its full feasible region, i.e., the feasible line segment. In each iteration of the binary line search, we set θ as the midpoint of its upper bound and lower bound values, $\theta = (V_u + V_\ell)/2$. We then compute the subgradient of $f(\theta)$ at this current point θ. Following Danskin's theorem, the subgradient of $f(\theta)$ can be computed as

$$\frac{\partial f}{\partial \theta} = \frac{\gamma}{2} \frac{\partial \mathrm{tr}([M^*, Z^*] E_\theta S^{*-1} E_\theta [M^*, Z^*]^\top)}{\partial \theta} \tag{16}$$

where M^*, Z^*, S^{*-1} are the optimal solution for the convex minimization problem in (15) with the given θ value. Since $f(\theta)$ is concave in θ, a positive subgradient value at θ indicates that the optimal θ^* value is larger than the current θ value, while a negative subgradient value indicates that the optimal θ^* value is smaller than the current θ value. Therefore, we increase the lower bound of θ to its current value when $\frac{\partial f}{\partial \theta} > 0$, and reduce the upper bound of θ to its current value when $\frac{\partial f}{\partial \theta} < 0$; thus ensuring the search interval is halved at each iteration.

By repeating the binary line search step, the feasible subinterval containing the optimal θ^* value can be quickly reduced at an exponential rate. When the subgradient is close to 0 or the interval between upper and lower bound values is sufficiently small, an optimal θ value can be returned. The overall binary search procedure is described in Algorithm 1.

4.2 Block-Coordinate Descent for Inner Convex Minimization

Both the computation of each subgradient value of $f(\theta)$ in the binary line search procedure and the final optimal solution recovery require solving the convex minimization problem in (15); i.e., the inner minimization problem in (13), for optimal M^*, Z^* and S^{*-1} given a fixed θ value. Although this optimization problem is convex, it is nevertheless challenging to design an efficient and scalable optimization algorithm to tackle the typically large parameter matrices M, Z, and S. For example, even for a given S, the Hessian matrix for the quadratic programming problem in M and Z can be too large to fit in memory for even a medium-sized data set with large number of input features.

Therefore, we develop a scalable block-descent optimization algorithm to solve the convex minimization problem iteratively. Specifically, in each iteration, we conduct an optimization over each of the three sets of variables, $\{S\}$, $\{M\}$ and $\{Z, \boldsymbol{\xi}, \boldsymbol{\eta}\}$, in turn, given all other variables fixed. The optimization over each of the three sub-problems is conducted as follows.

Optimization over $\boldsymbol{Z}$. Given fixed S and M values, the minimization problem over the remaining variables $Z, \boldsymbol{\xi}, \boldsymbol{\eta}$ forms a standard quadratic program with linear constraints; i.e.,

$$\begin{aligned} \min_{Z,\boldsymbol{\xi},\boldsymbol{\eta}} \quad & \mathbf{1}^\top\boldsymbol{\xi} + \mathbf{1}^\top\boldsymbol{\eta} + \frac{\gamma}{2}\mathrm{tr}([M,Z]Q[M,Z]^\top) \\ \text{subject to} \quad & \boldsymbol{\xi}_i \geq 1 - Z_{il} \;\; \text{for } l \in Y_{i:}, \forall i = 1 \cdots t_\ell \\ & \boldsymbol{\eta}_i \geq 1 + Z_{i\bar{l}} \;\; \text{for } \bar{l} \in \bar{Y}_{i:}, \forall i = 1 \cdots t_\ell \\ & \boldsymbol{\xi} \geq 0, \boldsymbol{\eta} \geq 0 \end{aligned} \tag{17}$$

where $Q = E_\theta S^{-1} E_\theta$. Note that the linear constraints are only expressed in terms of the labeled part of Z and $\boldsymbol{\xi}$, $\boldsymbol{\eta}$. It is easy to see that

$$\begin{aligned} \mathrm{tr}([M,Z]Q[M,Z]^\top) &= \mathrm{tr}([M,Z]\begin{bmatrix} Q_{dd} & Q_{dL} \\ Q_{Ld} & Q_{LL} \end{bmatrix}[M,Z]^\top) \\ &= \mathrm{tr}(MQ_{dd}M^\top + 2ZQ_{Ld}M^\top + ZQ_{LL}Z^\top) \end{aligned} \tag{18}$$

where Q_{dd} denotes the $d \times d$ top-left submatrix of Q, Q_{LL} denotes the $L \times L$ bottom-right submatrix of Q, and $Q_{dL} = Q_{Ld}^\top$ denotes the other two submatrices of Q. Moreover, it is known that the matrix Z and matrix M can both be decomposed into two submatrices corresponding to the labeled and unlabeled data, $Z = [Z^l; Z^u]$ and $M = [M^l; M^u]$. Thus (18) can be further rewritten as

$$\begin{aligned} (18) = \;& \mathrm{tr}(Z^\ell Q_{LL} Z^{\ell\top}) + 2\mathrm{tr}(Z^\ell Q_{Ld} M^{\ell\top}) + \mathrm{tr}(MQ_{dd}M^\top) \\ & + \mathrm{tr}(Z^u Q_{LL} Z^{u\top}) + 2\mathrm{tr}(Z^u Q_{Ld} M^{u\top}) \end{aligned} \tag{19}$$

which clearly shows that the optimization over submatrices Z^ℓ and Z^u can be conducted independently.

By setting the derivative of the objective (17) (which is also the derivative of (19)) with respect to Z^u to 0, we can obtain a closed-form solution for Z^u:

$$Z^u = -M^u Q_{dL} Q_{LL}^{-1}. \tag{20}$$

Although no closed-form solution exists for Z^ℓ due to the linear constraints in (17), note that the objective in (19) actually can be further decomposed into independent terms for each row of Z^ℓ. Furthermore, the linear constraints in (17) are row-wise separable regarding $Z^\ell, \boldsymbol{\xi}, \boldsymbol{\eta}$ as well. Therefore, we can optimize each row of Z^ℓ independently by solving a small standard quadratic programming. For example, the ith row of Z, $Z_{i:}$ and $\boldsymbol{\xi}_i, \boldsymbol{\eta}_i$, can be optimized as

$$\begin{aligned} \min_{Z_{i:},\boldsymbol{\xi}_i,\boldsymbol{\eta}_i} \quad & \boldsymbol{\xi}_i + \boldsymbol{\eta}_i + \frac{\gamma}{2} Z_{i:} Q_{LL} Z_{i:}^\top + \gamma Z_{i:} Q_{Ld} M_{i,:}^\top \\ \text{subject to} \quad & \boldsymbol{\xi}_i \geq 1 - Z_{il} \;\; \text{for } l \in Y_{i:}, \\ & \boldsymbol{\eta}_i \geq 1 + Z_{i\bar{l}} \;\; \text{for } \bar{l} \in \bar{Y}_{i:}, \\ & \boldsymbol{\xi}_i \geq 0, \boldsymbol{\eta}_i \geq 0 \end{aligned} \tag{21}$$

Algorithm 1. Binary Line Search

```
Input: $X, Y, \alpha, \beta, \gamma$, a small constant $\tau > 0$.
Initialize: set $V_\ell = 0, V_u = 1$.
Repeat:
    1. set $\theta = \frac{V_\ell + V_u}{2}$.
    2. given the current $\theta$, solve the inner minimization problem (15)
       for $M^*, Z^*, S^{*-1}$ using block-coordinate descent method.
    3. compute the subgradient $\frac{\partial f}{\partial \theta}$ according to Eq.(16).
    4. if $\|\frac{\partial f}{\partial \theta \|} < \tau$ then  return end if
    5. if $\frac{\partial f}{\partial \theta} > 0$ then  $V_\ell = \theta$ else  $V_u = \theta$ end if
Until $(V_u - V_\ell) < \tau$
```

Algorithm 2. Block-Coordinate Descent Optimization

```
Input: $X, Y, \alpha, \beta, \gamma$, a small constant $\tau > 0$.
Initialize: set $M = X$ and randomly initialize $Z$.
Repeat:
    1. recompute $S$ using Eq.(24).
    2. with given $S, M$, recompute $Z^u$ using Eq.(20), and recompute
       each row $Z^\ell_{i:}$ by solving the quadratic programming in (21).
    3. with given $S, Z$, recompute $M$ using Eq.(23).
Until changes in $M, Z$ is smaller than $\tau$.
```

which can be solved using any standard quadratic program solver.

Optimization over $\boldsymbol{M}$. Given fixed Z, $\boldsymbol{\xi}$, $\boldsymbol{\eta}$ and S, we optimize M as an unconstrained quadratic optimization problem

$$M = \arg\min_{M} \frac{\beta}{2}\|X - M\|_F^2 + \frac{\gamma}{2}\mathrm{tr}([M, Z]Q[M, Z]^\top). \tag{22}$$

Setting the derivative of the objective function with respect to M to 0 yields

$$M = (\beta X - \gamma Z Q_{Ld})(\beta I + \gamma Q_{dd})^{-1}. \tag{23}$$

Optimization over $\boldsymbol{S}$. Given Z, $\boldsymbol{\xi}$, $\boldsymbol{\eta}$ and M, the minimization over S has a closed-form solution as suggested in the Proposition we presented above; i.e.,

$$S = (E_\theta [M, Z]^\top [M, Z] E_\theta + \epsilon_i I)^{1/2} \tag{24}$$

where $\epsilon_i > 0$ is a small value added to achieve an invertible S.

The overall block-coordinate descent procedure is given in Algorithm 2. By employing the block-coordinate descent inner convex minimization, the binary line search algorithm we developed obviously provides a scalable optimization tool for the target non-smooth convex optimization problem.

Table 1. Data set properties: k = # classes, T = # instances, C = label cardinality

	Arts	Comp.	Edu.	Entain.	Health	Recr.	Ref.	Sci.	Soc.	Socie.
k	11	12	8	9	9	11	10	7	7	14
T	2525	2046	2816	2649	2932	2454	795	951	1181	4559
C	2.54	2.74	2.54	2.53	2.18	2.51	2.16	2.52	2.29	2.89

5 Experiments

To evaluate the proposed method, we conducted experiments on a set of multi-topic web page classification data sets [31]. Each data set consists of web pages collected from the yahoo.com domain. We preprocessed the data sets by first removing the largest class label (which covered more than 50% of the instances) and removing class labels that had fewer than 250 instances (for some data sets, we even used larger thresholds 300 and 400 to obtain larger label cardinalities). When the label cardinality of a data set is close to 1, the classification task is close to a standard single label multi-class task. The effectiveness of multi-label learning can be best demonstrated on data sets whose label cardinalities are reasonably large. We also removed any instances that had no labels or every label. For the input feature representation, we removed the less frequent features and converted the remaining integer features into a standard *tf-idf* encoding. The properties of the preprocessed data sets are summarized in Table 1.

We compared our proposed method (referred to as *TRANS* in the results) with four other large margin multi-label learning baselines:

- *CSRL*, the large margin multi-label learning method developed in (2) [8], based on a calibrated separation ranking loss.
- *CONS*, a variant of CSRL that replaces the regularizer $\|W\|_F^2$ with the constraint $\|W_{i:}\|_2 \leq \alpha$, as in Section 3.1.
- *dCSRL*, which first uses PCA to reduce the input dimension of the combined labeled and unlabeled data, then applies the CSRL method.
- *dCONS*, which first uses PCA to reduce the input dimension of the combined labeled and unlabeled data, then applies the CONS method.

Although numerous multi-label learning methods appear in the literature we restrict our attention to *convex* training methods to ensure that the results are repeatable independent of any particular implementation. In particular, for supervised multi-label losses we focus our comparison on CSRL, since previous work has demonstrated that this obtains state-of-the-art performance among convex supervised approaches [8]. (Note that the semi-supervised formulation presented in this paper can be easily applied to *any* convex multi-label loss [23]. However, finding tractable convex reformulations for losses specifically tailored for multi-labeled classification, such as F-measure [4], remains an open problem in the literature—e.g., the advanced formulation given in [4] still relies on an NP-hard constraint generation oracle.)

Table 2. Average transductive micro-F1 results over 10 repeats (± standard deviation)

Data set	CSRL	CONS	TRANS	dCSRL	dCONS
Arts	0.42 ±0.01	0.37 ±0.01	0.50 ±0.01	0.44 ±0.01	0.42 ±0.01
Computers	0.45 ±0.02	0.34 ±0.01	0.53 ±0.01	0.42 ±0.01	0.41 ±0.02
Education	0.56 ±0.02	0.45 ±0.01	0.61 ±0.01	0.56 ±0.01	0.49 ±0.02
Entertainment	0.61 ±0.03	0.44 ±0.02	0.63 ±0.01	0.50 ±0.02	0.46 ±0.03
Health	0.60 ±0.02	0.35 ±0.01	0.64 ±0.01	0.49 ±0.01	0.44 ±0.01
Recreation	0.48 ±0.02	0.32 ±0.05	0.53 ±0.01	0.41 ±0.01	0.39 ±0.01
Reference	0.55 ±0.02	0.41 ±0.01	0.55 ±0.01	0.36 ±0.01	0.34 ±0.01
Science	0.68 ±0.01	0.54 ±0.04	0.72 ±0.01	0.64 ±0.01	0.56 ±0.01
Social	0.63 ±0.01	0.53 ±0.06	0.67 ±0.01	0.55 ±0.01	0.48 ±0.01
Society	0.31 ±0.02	0.29 ±0.01	0.43 ±0.01	0.34 ±0.01	0.31 ±0.01

Table 3. Average transductive macro-F1 results on 10 repeats (± standard deviation)

Data set	CSRL	CONS	TRANS	dCSRL	dCONS
Arts	0.37 ±0.01	0.34 ±0.02	0.47 ±0.01	0.43 ±0.01	0.41 ±0.02
Computers	0.39 ±0.02	0.28 ±0.01	0.48 ±0.02	0.40 ±0.01	0.40 ±0.02
Education	0.48 ±0.02	0.39 ±0.01	0.54 ±0.01	0.54 ±0.01	0.47 ±0.02
Entertainment	0.50 ±0.05	0.34 ±0.01	0.53 ±0.03	0.46 ±0.01	0.42 ±0.03
Health	0.52 ±0.02	0.31 ±0.01	0.57 ±0.01	0.47 ±0.01	0.42 ±0.01
Recreation	0.37 ±0.02	0.28 ±0.02	0.45 ±0.01	0.38 ±0.01	0.36 ±0.01
Reference	0.34 ±0.01	0.32 ±0.01	0.42 ±0.01	0.32 ±0.01	0.30 ±0.01
Science	0.62 ±0.02	0.47 ±0.04	0.67 ±0.01	0.61 ±0.01	0.54 ±0.01
Social	0.53 ±0.02	0.44 ±0.05	0.58 ±0.02	0.51 ±0.01	0.45 ±0.01
Society	0.19 ±0.02	0.24 ±0.01	0.34 ±0.01	0.32 ±0.01	0.29 ±0.01

To also provide a comparison to *semi-supervised* methods, along the lines of [19–21], we furthermore include the latter two competitors, which use the unlabeled and labeled data to first learn a low dimensional representation for the input data. Note that the dimensionality reduction in this case is independent of the target labels. The goal of these experiments therefore is to isolate the consequences of using unlabeled data for subspace identification, and using label information in choosing such subspaces.

In these experiments we simply set the regularization parameters for TRANS to $\alpha = 0.01$, $\beta = 100$ and $\gamma = 50$, and set the regularization parameter for CSRL and CONS to $\alpha = 0.01$. The target dimensionality was set to 50 for the dimensionality reduction methods dCSRL and dCONS. The performance of each method is evaluated using the macro-F1 and micro-F1 measures [32]. We randomly selected 200 instances from each data set to be the labeled part, and another 1000 instances to be the unlabeled part. The process is repeated 10 times to generate 10 random partitions. The average performance and standard deviations of the five methods are reported in Table 2 and Table 3 respectively.

One can see from these results that the unlabeled data generally provides an improvement in generalization accuracy over the baseline supervised methods. First, using dimensionality reduction as a preprocessing step only gave mixed benefits for the CSRL and CONS methods, although CONS generally benefited more. Interestingly, the proposed TRANS method, which learns a low dimensional subspace that also depends on the training labels, obtains a systematic and noticeable improvement over the other methods. TRANS significantly improves the macro-F1 measure over all comparison methods in every case, while achieving the same result for micro-F1 measure in every case except the "Reference" data set. The run times of TRANS and CSRL on 1200 data points (200 labeled and 1000 unlabeled) were approximately 10m for TRANS and 1m for CSRL, respectively, using simple Matlab implementations.

6 Conclusions

We have proposed a new method for semi-supervised multi-label classification that combines a state-of-the-art large margin multi-label learning approach with a current representation learning method. A key aspect of this approach is that it allows an efficient global training procedure. Experimental results show that the semi-supervised combination can outperform corresponding supervised and simple semi-supervised learning methods in a transductive setting.

There remains several important directions for future work. The current formulation is transductive; an out-of-sample extension of our approach is possible using a proposed technique from [23]. It also remains to investigate other representation learning formulations, such as $p = 1$, to determine their impact on performance. Another interesting direction is to extend the work of [4] to incorporate a tractable convex relaxation of F-measure for training.

References

1. Joachims, T.: Text Categorization with Support Vector Machines: Learn with Many Relevant Features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, Springer, Heidelberg (1998)
2. McCallum, A.: Multi-label text classification with a mixture model trained by em. In: AAAI Workshop on Text Learning (1999)
3. Zhu, S., Ji, X., Xu, W., Gong, Y.: Multi-labelled classification using maximum entropy method. In: Conference on Information Retrieval, SIGIR (2005)
4. Petterson, J., Caetano, T.: Submodular multi-label learning. In: Advances in Neural Information Processing Systems, NIPS (2011)
5. Kazawa, H., Izumitani, T., Taira, H., Maeda, E.: Maximal margin labeling for multi-topic text categorization. In: Neural Infor. Processing Sys., NIPS (2004)
6. Godbole, S., Sarawagi, S.: Discriminative Methods for Multi-labeled Classification. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 22–30. Springer, Heidelberg (2004)
7. Hariharan, B., Zelnik-Manor, L., Vishwanathan, S., Varma, M.: Large scale max-margin multi-label classification with priors. In: Proceedings ICML (2010)
8. Guo, Y., Schuurmans, D.: Adaptive large margin training for multilabel classification. In: Conference on Artificial Intelligence, AAAI (2011)
9. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in Neural Information Processing, NIPS (2001)

10. Schapire, R., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning 39(2-3), 135–168 (2000)
11. Shalev-Shwartz, S., Singer, Y.: Efficient learning of label ranking by soft projections onto polyhedra. Journal of Machine Learning Research 7, 1567–1599 (2006)
12. Fuernkranz, J., Huellermeier, E., Mencia, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning 73(2) (2008)
13. Yu, K., Yu, S., Tresp, V.: Multi-label informed latent semantic indexing. In: Conference on Research and Development in Information Retrieval, SIGIR (2005)
14. Yan, R., Tesic, J., Smith, J.: Model-shared subspace boosting for multi-label classification. In: Conference on Knowledge Discovery and Data Mining, KDD (2007)
15. Zhang, M., Zhou, Z.: Multi-label dimensionality reduction via dependency maximization. In: Conference on Artificial Intelligence, AAAI (2008)
16. Rai, P., Daumé III, H.: Multi-label prediction via sparse infinite CCA. In: Advances in Neural Information Processing Systems, NIPS (2009)
17. Kong, X., Yu, P.: Multi-label feature selection for graph classification. In: Proc. of the IEEE International Conference on Data Mining, ICDM (2010)
18. Ji, S., Tang, L., Yu, S., Ye, J.: A shared-subspace learning framework for multi-label classification. ACM Trans. Knowl. Discov. Data 4(2), 1–29 (2010)
19. Liu, Y., Jin, R., Yang, L.: Semi-supervised multi-label learning by constrained non-negative matrix factorization. In: Conf. on Artificial Intelligence, AAAI (2006)
20. Chen, G., Song, Y., Wang, F., Zhang, C.: Semi-supervised multi-label learning by solving a sylvester equation. In: SIAM Conference on Data Mining, SDM (2008)
21. Qian, B., Davidson, I.: Semi-supervised dimension reduction for multi-label classification. In: Conference on Artificial Intelligence, AAAI (2010)
22. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In: Advances in Neural Information Processing Systems, NIPS (2006)
23. Zhang, X., Yu, Y., White, M., Huang, R., Schuurmans, D.: Convex sparse coding, subspace learning, and semi-supervised extensions. In: Conference on Artificial Intelligence, AAAI (2011)
24. Lee, H., Battle, A., Raina, R., Ng, A.: Efficient sparse coding algorithms. In: Advances in Neural Information Processing Systems, NIPS (2006)
25. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. Machine Learning 73, 243–272 (2008)
26. Bach, F., Mairal, J., Ponce, J.: Convex sparse matrix factorizations. arXiv:0812.1869v1 (2008)
27. Lee, H., Raina, R., Teichman, A., Ng, A.: Exponential family sparse coding with application to self-taught learning. In: Int. Joint Conf. Artif. Intell., IJCAI (2009)
28. Rish, I., Grabarnik, G., Cecchi, G., Pereira, F., Gordon, G.: Closed-form supervised dimensionality reduction with generalized linear models. In: Proc. ICML (2007)
29. Hendrickx, J., Olshevsky, A.: Matrix p-norms are NP-hard to approximate if $p \neq 1, 2, \infty$. SIAM J. Matrix Anal. Appl. 31(5), 2802–2812 (2010)
30. Grave, E., Obozinski, G., Bach, F.: Trace lasso: a trace norm regularization for correlated designs. In: Neural Information Processing Systems, NIPS (2011)
31. Ueda, N., Saito, K.: Parametric mixture models for multi-labeled text. In: Advances in Neural Information Processing Systems, NIPS (2002)
32. Tang, L., Rajan, S., Narayanan, V.: Large scale multi-label classification via metalabeler. In: International WWW Conference (2009)

MDL-Based Analysis of Time Series at Multiple Time-Scales

Ugo Vespier[1], Arno Knobbe[1], Siegfried Nijssen[2], and Joaquin Vanschoren[1]

[1] LIACS, Leiden University, The Netherlands
[2] Katholieke Universiteit Leuven, Belgium
uvespier@liacs.nl

Abstract. The behavior of many complex physical systems is affected by a variety of phenomena occurring at different temporal scales. Time series data produced by measuring properties of such systems often mirrors this fact by appearing as a composition of signals across different time scales. When the final goal of the analysis is to model the individual phenomena affecting a system, it is crucial to be able to recognize the right temporal scales and to separate the individual components of the data. In this paper, we approach this challenge through a combination of the Minimum Description Length (MDL) principle, feature selection strategies, and convolution techniques from the signal processing field. As a result, our algorithm produces a good decomposition of a given time series and, as a side effect, builds a compact representation of its identified components. Experiments demonstrate that our method manages to identify correctly both the number and the temporal scale of the components for real-world as well as artificial data and show the usefulness of our method as an exploratory tool for analyzing time series data.

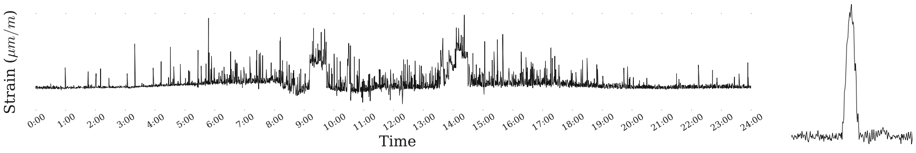
Keywords: Time Series, Scale Selection, Minimum Description Length.

1 Introduction

This paper is concerned with the analysis of sensor data. When monitoring complex physical systems over time, one often finds multiple phenomena in the data that work on different time scales. If one is interested in analyzing and modeling these individual phenomena, it is crucial to recognize these different scales and separate the data into its underlying components. Here, we present a method for extracting the time scales of various phenomena present in large time series. The method combines concepts from the signal processing domain with feature selection and the Minimum Description Length principle [2].

The need for analyzing time series data at multiple time scales is nicely demonstrated by a large monitoring project in the Netherlands, called *InfraWatch* [6,11]. In this project, we employ a range of sensors to measure the dynamic response of a large Dutch highway bridge to varying traffic and weather conditions. When viewing this data (see Fig. 1a), one can easily distinguish various *transient events* in the signal that occur on different time scales. Most notable

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 371–386, 2012.



(a) (b)

Fig. 1. (a) One day of strain measurements from a large highway bridge in the Netherlands. The multiple external factors affecting the bridge are visible at different time scales. (b) A detail of plot (a) showing one of the peaks caused by passing vehicles.

are the gradual change in strain over the course of the day (as a function of the outside temperature, which influences stiffness parameters of the concrete), a prolonged increase in strain caused by rush hour traffic congestion, and individual bumps in the signal due to cars and trucks traveling over the bridge. In order to understand the various changes in the sensor signal, one would benefit substantially from separating out the events at various scales. The main goal of the work described here is to do just that: we consider the temporal data as a series of superimposed effects at different time scales, establish at which scales events most often occur, and from this we extract the underlying signal components.

In this work, we approach the scale selection problem from a Minimum Description Length (MDL) perspective (see Section 3). The motivation for this is that we need a framework in which we can deal with a wide variety of representations for scale components. The MDL framework was shown to be sufficiently general to provide this flexibility by Hu et al. [3] for the problem of choosing the best model for a given signal. Our main assumption here is that separating the original signal into components at different time scales will simplify the shape of the individual components, making it easier to model them separately. Our results show that, indeed, these multiple models outperform (in terms of MDL score) a single model derived from the original signal. While introducing multiple models incurs the penalty of having to describe these multiple models, there are much fewer 'exceptions' to be described compared to the single model, yielding a lower overall description length. For instance, in the sensor data of Fig. 1a, cars are often passing in one direction while there is rush hour congestion in the opposite direction. Using multiple models, this is modeled accurately, while a single model will easily ignore these events.

The analysis of time scales in time series data is often approached from a *scale-space* perspective, which involves convolution of the original signal with Gaussian kernels of increasing size [12] to remove information at smaller scales. By subtracting carefully selected components of the scale-space, we can effectively cut up the scale space into k ranges. In other words, signal processing offers methods for producing a large collection of derived features, and the challenge we face in this paper is how to select a subset of k features, such that the original signal is decomposed into a set of meaningful components at different scales.

Our approach applies the MDL philosophy to various aspects of modeling: choosing the appropriate scales at which to model the components, determining the optimal number of components (while avoiding overfitting on overly specific details of the data), and deciding which class of models to apply to each individual component. For this last decision, we propose two classes of models representing the components respectively on the basis of a discretization and a segmentation scheme. For this last scheme, we allow three levels of complexity to approximate the segments: piecewise constant approximations, piecewise linear approximations, as well as quadratic ones. These options result in different trade-offs between model cost and accuracy, depending on the type of signal we are dealing with.

A useful side product of our approach is that it identifies a concise representation of the original signal. This representation is useful in itself: queries run on the decomposed signal may be answered more quickly than when run on the original data. Furthermore, the parameters of the encoding may indicate useful properties of the data as well.

The paper is organized as follows. Section 2 reviews the signal processing concepts used in this work and introduces the concept of scale-space decomposition. Section 3 shows how we encode the signal decompositions and use MDL to select the best subset of scales. Section 4 presents an empirical evaluation of our method on both real-world and artificial data. Section 5 links our method to related work. Finally, Section 6 states our main conclusions and ideas for future work.

2 Preliminaries

In this section we introduce the notation and the basic definitions used throughout the paper. In particular, we review the concept of the scale-space image of a signal and we show how to exploit it to define a set of candidate scale-space decompositions. We deal with finite sequences of numerical measurements (samples), collected by observing some property of a system with a sensor, and represented in the form of time series as defined below.

Definition 1. *A* ***time series*** *of length n is a finite sequence of values $\boldsymbol{x} = x[1], \ldots, x[n]$ of finite precision.*[1] *A subsequence $\boldsymbol{x}[a:b]$ of $\boldsymbol{x}$ is defined as follows:*

$$\boldsymbol{x}[a:b] = (\boldsymbol{x}[a], \boldsymbol{x}[a+1], \ldots, \boldsymbol{x}[b]),\ a < b$$

We also assume that all the considered time series have no missing values and that their sampling rate is constant.

2.1 The Scale-Space Image

The *scale-space image* [12] is a scale parametrization technique for one-dimensional signals[2] based on the operation of convolution.

[1] 32-bit floating point values in our experiments.

[2] From now on, we will use the term signal and time series interchangeably.

Definition 2. *Given a signal $\boldsymbol{x}$ of length n and a response function (kernel) $\boldsymbol{h}$ of length m, the result of the* ***convolution*** *$\boldsymbol{x} * \boldsymbol{h}$ is the signal $\boldsymbol{y}$ of length n, defined as:*

$$\boldsymbol{y}[t] = \sum_{j=-m/2+1}^{m/2} \boldsymbol{x}[t-j]\, \boldsymbol{h}[j]$$

In this paper, $\mathbf{h}$ is a Gaussian kernel with mean $\mu = 0$, standard deviation σ, area under the curve equal to 1, discretized into m values.[3] Also, since $\mathbf{x}$ is finite, $\mathbf{x}[t-j]$ may be undefined. To account for these boundary effects, $\mathbf{x}$ is padded with $m/2$ zeros before and after its defined range. A complete overview on how to compute the Gaussian convolutions for discrete signals can be found in [7].

The convolution acts as a *smoothing filter* which smooths each value $\mathbf{x}[t]$ based on its surrounding values. The amount of removed detail is directly proportional to the standard deviation σ (and thus m), from now on referred to as the *scale parameter*. In the limit, when $\sigma \to \infty$, the result of the Gaussian convolution converges to the mean of the signal $\mathbf{x}$.

Given a signal $\mathbf{x}$, the family of σ-smoothed signals $\Phi_{\mathbf{x}}$ over scale parameter σ is defined as follows:

$$\Phi_{\mathbf{x}}(\sigma) = \mathbf{x} * \mathbf{g}_{\sigma}\, , \ \sigma > 0$$

where $\mathbf{g}_{\sigma}$ is a Gaussian kernel having standard deviation σ, and $\Phi_{\mathbf{x}}(0) = \mathbf{x}$.

The signals in $\Phi_{\mathbf{x}}$ define a surface in the time-scale plane (t, σ) known in the literature as the *scale-space image* [7,12]. This visualization gives a complete description of the scale properties of a signal in terms of Gaussian smoothing. Moreover, it has other properties useful for segmentation, as we will see later in the paper.

For practical purposes, the scale-space image is quantized across the scale dimension by computing the convolutions only for a finite number of scale parameters. More formally, for a given signal $\mathbf{x}$, we fix a set of scale parameters

$$S = \{2^i \mid 0 \leq i \leq \sigma_{max} \ \wedge i \in \mathbb{N}\}$$

and we compute $\Phi_{\mathbf{x}}(\sigma)$ only for $\sigma \in S$ where σ_{max} is such that $\Phi_{\mathbf{x}}(\sigma)$ is approximately equal to the mean signal of $\mathbf{x}$.

As an example, Figure 2 shows the scale-space image of an artificially generated signal. The topmost plot represents the original signal, constructed by three components at different temporal scales: a slowly changing and slightly curved baseline, medium term events (bumps) and short term events (peaks). It is easy to visually verify that, by increasing the scale parameter, a larger amount of detail is removed. In particular, the peaks are smoothed out at scales greater than $\sigma = 2^4$, and the bumps are smoothed out at scales greater than $\sigma = 2^8$, after which only the baseline remains.

In the next section, we show how to manipulate the scale-space image to filter out the effects of transient events in a specific range of scales. This will lead to the definition of a signal decomposition scheme.

[3] To capture almost all non-zero values, we define $m = \lfloor 6\sigma \rfloor$.

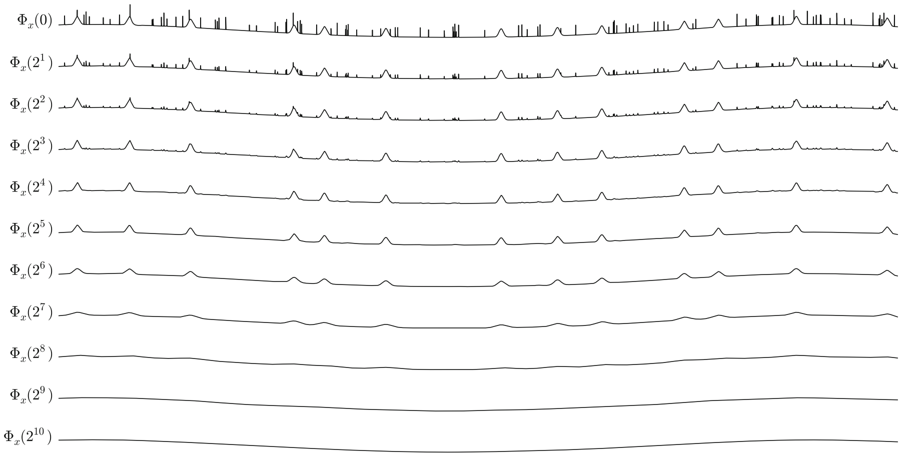


Fig. 2. Scale-space image of an artificially generated signal totalling 259200 points

2.2 Scale-Space Decomposition

Along the scale dimension of the scale-space image, short-time transient events in the signal will be smoothed away sooner than longer ones. In other words, we can associate to each event a maximum scale σ_{cut} such that, for $\sigma > \sigma_{cut}$, the transient event is no longer present in $\Phi_{\mathbf{x}}(\sigma_{cut})$. This fact leads to the following two observations:

- Given a signal scale-space image $\Phi_{\mathbf{x}}$, the signal $\Phi_{\mathbf{x}}(\sigma)$ is only affected by the transient events at scales greater than σ. This is conceptually equivalent to a *low-pass filter* in signal processing.
- Given a signal scale-space image $\Phi_{\mathbf{x}}$ and two scales $\sigma_1 < \sigma_2$, the signal $\Phi_{\mathbf{x}}(\sigma_1) - \Phi_{\mathbf{x}}(\sigma_2)$ is mostly affected by those transient events present in the range of scales (σ_1, σ_2). This is similar to a *band-pass filter* in signal processing.

As an example, reconsider the signal $\mathbf{x}$ and its scale-space image $\Phi_{\mathbf{x}}$ of Figure 2. Figure 3 shows (from top to bottom):

- the signal $\Phi_{\mathbf{x}}(0) - \Phi_{\mathbf{x}}(2^4)$, which is the result of a high-pass filtering; this feature represents the short-term events (peaks),
- the signal $\Phi_{\mathbf{x}}(2^4) - \Phi_{\mathbf{x}}(2^{10})$, which is the result of a band-pass filtering; this feature represents the medium-term events (bumps),
- the signal $\Phi_{\mathbf{x}}(2^{10})$, which is the result of a low-pass filtering; this feature represents the long-term trend.

Generalizing the example in Figure 3, we can define a decomposition scheme of a signal $\mathbf{x}$ by considering adjacent ranges of scales of the signal scale-space image. We formalize this idea below.

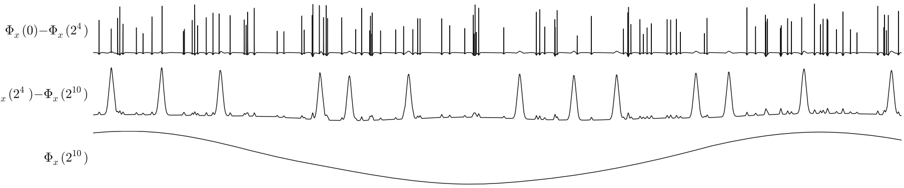


Fig. 3. Examples of signal decomposition obtained from the scale-space image in Figure 2

Definition 3. *Given a signal* $\boldsymbol{x}$ *and a set of* $k-1$ *scale parameters* $C = \{\sigma_1, \ldots, \sigma_{k-1}\}$ *(called the cut-points set) such that* $\sigma_1 < \ldots < \sigma_{k-1}$*, the* ***scale decomposition*** *of* $\boldsymbol{x}$ *is given by the set of component signals* $D_{\boldsymbol{x}}(C) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}$*, defined as follows:*

$$\boldsymbol{x}_i = \begin{cases} \Phi_{\boldsymbol{x}}(0) - \Phi_{\boldsymbol{x}}(\sigma_1) & \textit{if } i = 1 \\ \Phi_{\boldsymbol{x}}(\sigma_{i-1}) - \Phi_{\boldsymbol{x}}(\sigma_i) & \textit{if } 1 < i < k \\ \Phi_{\boldsymbol{x}}(\sigma_{k-1}) & \textit{if } i = k \end{cases}$$

Note that for k components we require $k-1$ cut-points. This decomposition has several elegant properties:

- $\mathbf{x}_k$ can be seen as the baseline of the signal, as obtained by a low-pass filter;
- $\mathbf{x}_i$ for $1 \leq i < k$ are signals as obtained by a band-pass filter, and can be used to identify transient events;
- $\sum_{i=1}^{k} \mathbf{x}_i = \mathbf{x}$, i.e., the original signal can be recovered from the decomposition.

3 MDL Scale Decomposition Selection

Given an input signal $\mathbf{x}$, the main computational challenge we face is twofold:

- find a good subset of cut-points C such that the resulting k components of the decomposition $D_{\mathbf{x}}(C)$ optimally capture the effect of transient events at different scales,
- select a representation for each component, according to its inherent complexity.

As stated before, the rationale behind the scale decomposition is that it is easier to model the effect of a single class of transient events at a given scale than to model the superimposition of many, interacting transient events at multiple scales. We thus need to trade off the added complexity of having to represent multiple components for the complexity of the representations themselves. In this paper, we propose to use the Minimum Description Length (MDL) principle to approach this problem.

The Minimum Description Length [2] is an information-theoretic model selection framework that selects the best model according to its ability to *compress* the given data. In our context, the two-part MDL principle states that the best model M to describe the signal $\mathbf{x}$ is the one that minimizes the sum $L(M) + L(\mathbf{x} \mid M)$, where

- $L(M)$ is the length, in bits, of the description of the model,
- $L(\mathbf{x} \mid M)$ is the length, in bits, of the description of the signal when encoded with the help of the model M.

The possible models depend on the scale decomposition $D_{\mathbf{x}}(C)$ considered[4] and on the representations used for its individual components. An ideal set of representations would adapt to the specific features of every single component, resulting in a concise summarization of the decomposition and, thus, of the signal. In order to apply the MDL principle, we need to define a model $M_{D_{\mathbf{x}}(C)}$ for a given scale decomposition $D_{\mathbf{x}}(C)$ and, consequently, how to compute both $L(M_{D_{\mathbf{x}}(C)})$ and $L(\mathbf{x} \mid M_{D_{\mathbf{x}}(C)})$. The latter term is the length in bits of the information lost by the model, i.e., the residual signal $\mathbf{x} - M_{D_{\mathbf{x}}(C)}$.

As the MDL framework is only applicable to discrete data, we first clarify below how we discretize the input signal $\mathbf{x}$ and all the subsequent operations. Subsequently, we will introduce the proposed representation schemes for the components and define the bit complexity of the residual and the model selection procedure.

3.1 Time Series Values Discretization

In order to use the MDL principle we need to work with a quantized input signal and scale-space image. Because of this, we assume that the values v of both the input signal $\mathbf{x}$ and $\Phi_{\mathbf{x}}(\sigma)$, for each considered σ, have been quantized to a finite number of symbols by employing the function defined below:

$$Q(v) = \left\lfloor \frac{v - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}\, l \right\rfloor - \frac{l}{2}$$

where l, assumed to be even, is the number of bins to use in the discretization while $\min(\mathbf{x})$ and $\max(\mathbf{x})$ are respectively the minimum an maximum value in $\mathbf{x}$. Throughout the rest of the paper, we assume $l = 256$. A similar approach is described in [3]. All the subsequent operations, from the computations of the scale decompositions to the encoding of the components, are kept in this quantized space.

3.2 Component Representation Schemes

Within our general framework, many different approaches could be used for representing the components of a decomposition. In the next paragraphs we introduce two such methods.

[4] Including the decomposition formed by zero cut-points ($C = \emptyset$), i.e., the signal itself.

Discretization-Based Representation. In some components of our data transient events always occur with similar amplitudes, mixed with long stretches of baseline values (see Figure 3). Hence, a desirable encoding could be one that captures this repetitiveness in the data by giving short codes to long stretches of the baseline and the commonly occurring amplitudes. Unfortunately, our original discretization is too fine-grained to capture regular occurrences of similar amplitudes. As a first representation, we hence propose to also consider more coarse-grained discretizations of the original range of values. We do this by discretizing each value v in a component to a value $\lfloor Q(v)/2^i \rfloor$, where several values for i are considered for each component, typically $i \in \{2, 4, 6\}$. By doing so, similar values will be grouped together in the same bin. The resulting sequence of integers is compacted further by performing run-length encoding, resulting in a string of (v, l) pairs, where l represents the number of times value v is repeated consecutively. This string is finally encoded using a Shannon-Fano or Huffman code (see Section 3.3).

As a simplified illustration of how the MDL principle helps here to identify components, consider data generated by the expression $(67)^n(01)^n$ ($4n$ integers from the range $\{0, \ldots, 2^3 - 1\}$), where we assume n and the range are fixed. In this data, each symbol occurs with the same frequency; we can encode the time series hence with $-\log_2(1/4) \cdot 4 \cdot n = 8n$ bits for the data, plus $8 \log n$ bits for the dictionary of frequencies. Consider now the decomposition of the signal into two time series, $6^{2n}0^{2n}$ and $(01)^{2n}$. The first component, of which the run-length encoding is $(6, 2n)(0, 2n)$, can be encoded using only 2 bits for the time series (as there is only one possible run-length value, we use 0 bits to encode the run-lengths), $8 \log n$ bits for the dictionary of amplitudes, and $3 \log n$ bits to identify the length of the one run-length ($\log n$ bit for identifying the number of run-lengths, in this case one, $\log n$ to identify the one run-length present, and $\log n$ to identify its frequency, from which the encoding with 0 bits follows). The second component can be encoded using $4n$ bits for the time series, as well as $8 \log n$ bits for the dictionary. Assuming we also use 1 bit per component to identify the type of encoding used, this gives us an encoding in $4 + 19 \log n + 4n$ bits. Comparing this to $8n + 8 \log n$ bits, for $n \geq 11$ we will hence correctly identify the two components in this simplified data.

Segmentation-Based Representation. The main assumption on which we base this method is that a clear transient event can be accurately represented by a simple function, such as a polynomial of a bounded degree. Hence, if a signal contains a number of clear transient events, it should be possible to accurately represent this signal with a number of segments, each of which represented by a simple function.

Given a component $\mathbf{x}_i$ of length n, let

$$z(\mathbf{x}_i) = \{t_1, t_2, ..., t_m\}, \quad 1 < t_i \leq n$$

be a set of indexes of the segment boundaries.

Let $\texttt{fit}(\mathbf{x}_i[a:b], d_i)$ be the approximation of $\mathbf{x}_i[a:b]$ obtained by fitting a polynomial of degree d_i. Then, we represent each component $\mathbf{x}_i$ with the approximation $\hat{\mathbf{x}}_i$, such that:

$$\begin{aligned} \hat{\mathbf{x}}_i[0:z_1] &= \texttt{fit}(\mathbf{x}_i[0:z_1], d_i) \\ \hat{\mathbf{x}}_i[z_i:z_{i+1}] &= \texttt{fit}(\mathbf{x}_i[z_i:z_{i+1}], d_i),\ 1 \leq i < m \\ \hat{\mathbf{x}}_i[z_m:n] &= \texttt{fit}(\mathbf{x}_i[z_m:n], d_i) \end{aligned}$$

Note that approximation $\hat{\mathbf{x}}_i$ is quantized again by reapplying the function Q to each of its values.

For a given k-components scale decomposition $D_{\mathbf{x}}(C)$ and a fixed polynomial degree for each of its components, we calculate the complexity in bits of the model $M_{D_{\mathbf{x}}(C)}$, based on this representation scheme, as follows. Each approximated component $\hat{\mathbf{x}}_i$ consists of $|z(\mathbf{x}_i)|+1$ segments. For each segment, we need to represent its length and the d_i+1 coefficients of the fitted polynomial. The length ls_i of the longest segment in $\hat{\mathbf{x}}_i$ is given by

$$ls_i = \max(z_1 \cup \{z_{i+1} - z_i \mid 0 < i \leq m\})$$

We therefore use $\log_2(ls_i)$ bits to represent the segment lengths, while for the coefficients of the polynomials we employ floating point numbers of fixed[5] bit complexity c. The MDL model cost is thus defined as:

$$L(M_{D_{\mathbf{x}}(C)}) = \sum_{i=1}^{k} (|z(\mathbf{x}_i)|+1)\left(\lceil \log_2(ls_i) \rceil + c\,(d_i+1)\right)$$

So far we assumed to have a set of boundaries $z(\mathbf{x}_i)$, but we did not specify how to compute them. A desirable property for our segmentation would be that a segmentation at a coarser scale does not contain more segments than a segmentation at a finer scale.

The scale space theory assures that there are fewer zero-crossing of the derivatives of a signal at coarser scales [12]. In our segmentation we use the zero-crossings of the first and second derivatives.

More formally, we define the segmentation boundaries of a component $\mathbf{x}_i$ to be

$$z(\mathbf{x}_i) = \left\{ t \in \mathbb{R} \;\middle|\; \frac{d\mathbf{x}_i}{dt}(t) = 0 \right\} \bigcup \left\{ t \in \mathbb{R} \;\middle|\; \frac{d^2\mathbf{x}_i}{dt}(t) = 0 \right\}.$$

Figure 4b shows an example of segmentation obtained as above using fitted polynomials of degree 1.

However, many other segmentation algorithms are known in the literature [4,5] and all of them can be interchangeably employed in this context.

3.3 Residual Encoding

Given a model $M_{D_{\mathbf{x}}(C)}$, its residual $\mathbf{r} = \mathbf{x} - \sum_{i=1}^{k} \hat{\mathbf{x}}_i$, computed over the components approximations, represents the information of $\mathbf{x}$ not captured by the

[5] In our experiments $c = 32$.

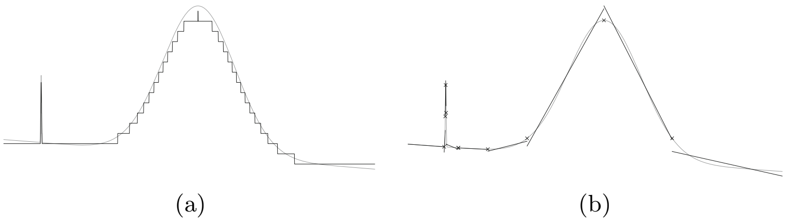


Fig. 4. Example of discretization-based encoding (a) and segmentation-based encoding with first degree polynomial approximations (the markers show the zero-crossings) (b)

model. Having already defined the model cost for the two proposed encoding schemes, we only still need to define $L(\mathbf{x} \mid M_{D_{\mathbf{x}}(C)})$, i.e., a bit complexity $L(\mathbf{r})$ for the residual $\mathbf{r}$.

Here, we exploit the fact that we operate in a quantized space; we encode each bin in the quantized space with a code that uses approximately $-\log(P(x))$ bits, where $P(x)$ is the frequency of the xth bin in our data. The main justification for this encoding is that we expect that the errors are normally distributed around 0. Hence, the bins in the discretization that reflect a low error will have the highest frequency of occurrences; we will give these the shortest codes. In practice, such codes can be obtained by means of Shannon-Fano coding or Huffman coding; as Hu et al. [3] we use Huffman coding in our experiments.

3.4 Model Selection

We can now define the MDL score that we are optimizing as follows:

Definition 4. *Given a model $M_{D_x(C)}$, its **MDL score** is defined as:*

$$L(M_{D_x(C)}) + L(\boldsymbol{r})$$

In the case of discretization-based encoding, the MDL score is affected by the cardinality used to encode each component. In the case of segmentation-based encoding the MDL score depends on the boundaries of the segments and the degrees of the polynomials in the representation. In both cases, also the cut-points of the considered decomposition affect the final score.

The simplest way to find the model that minimizes this score is to enumerate, encode and compute the MDL score for every possible scale-space decomposition and all possible encoding parameters. As we shall now show, this brute-force approach is practically feasible.

The number of possible scale decompositions depends on the total number of cut-points sets we can build from the computed scale parameters in $\Phi_{\mathbf{x}}$. We fix the maximum number of cut-points in a candidate set to some value c_{max}. This also means that we limit our search to those scale decompositions having $c_{max} + 1$ components or less. Moreover, given our wish to consider only simple approximations of the signals, we can also assume a reasonably low limit d_{max}

(in practice, $d_{max} = 2$) on the degree of the polynomials that approximate the segments of each given component.

Computing the MDL score for each encoded scale decomposition, obtained by ranging over all the possible configurations of cut-points $C_1, ..., C_{k-1}$, and all the possible configurations of polynomial degrees $d_1, ..., d_k$, hence requires calculating MDL scores for

$$\sum_{k=2}^{c_{max}+1} \binom{|\mathbf{S}|}{k-1} d_{max}^k$$

scale decompositions. This turns out to be a reasonable number in most practical cases we consider, and hence we use an exhaustive approach in our experiments.

4 Experiments

In this section, we experimentally evaluate our method, both on artificial data and on actual sensor data from the highway bridge mentioned in the introduction. To evaluate the strengths and weaknesses of our method, we have tested it on a range of artificial datasets[6] that mimic some of the multi-scale phenomena present in the bridge data. Our constructed data deliberately varies from easy, with clearly separated scales, to challenging with a variety of event shapes and sizes. All artificial datasets represent sensor data measured at 1 Hz for a duration of three days (totaling 259,200 data points). The data was produced by combining three components at three distinct scales, resembling 1) individual events from vehicles, 2) traffic jams that last several tens of minutes, and 3) gradual change of the baseline, due to temperature changes of the bridge over the course of several days.

Artificial Data. We start by considering one particular dataset in detail (see Figure 5a). This dataset was constructed by using Gaussian shapes for both the small and medium-scale events, and a sine wave of period 2.25 days at the largest scale. Medium events have a constant height, whereas small-scale events have a random height. We limited the search space to decompositions having a maximum of 4 components (3 cut-points). As can be seen in Figure 5a, our method was able to identify the fact that this data contains three important scales. Furthermore, the method correctly identified the two necessary cut-points, such that the three original components were reconstructed. The selected cut-points[7] appear at scales $2^9 = 512$ and $2^{12} = 4096$. When considering the separated components in detail, some influence across the scale-boundaries is visible, for example where small effects of the 'traffic jams' appear among the small-scale

[6] The artificial datasets and the source code can be obtained by contacting the first author.

[7] Note that our method returns the boundaries between scales, rather than the actual scales of the original components.

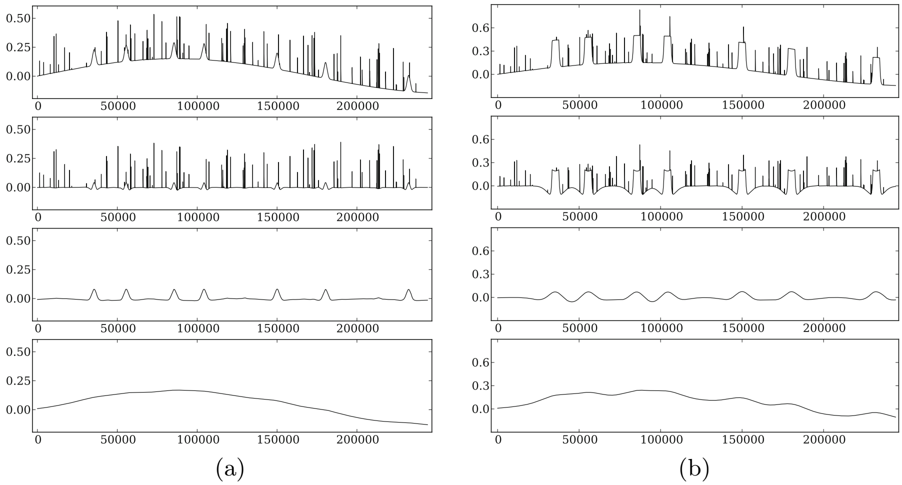


Fig. 5. Signals (top) and top-ranked decompositions for the two artificial datasets

events. These effects seem unavoidable, with the inherent limitations of the scale-space-based band-pass filtering and the discrete collection of scales we consider (powers of 2).

This optimal result has an MDL-score of 509,000 bits, being the sum of the model cost ($L(M) = 75,072$) and the error length ($L(D \mid M) = 433,928$). The second-ranked result on this data, with cut-points $C = \{2^{11}, 2^{13}\}$, shows a similar result, however with slightly more pronounced cross-boundary artifacts in the smallest scale, as is expected with a doubling of the lower cut-point. The MDL-score of this result is $64,896 + 450,487 = 515,383$. The $k = 1$ case, which corresponds to compression of the original signal without any decomposition, appears at rank three, with an MDL-score of $44,640 + 471,271 = 515,911$. This model obviously has a much lower model cost, due to having to represent only a single component, but this is compensated by the substantially higher error length, putting it below the scale-separated results. Ranks four and five represent two $k = 2$ results, where the former groups the small and medium scales together, and the latter the medium and large. All results in the top 10 relate to models that use polynomial representations ($d \leq 2$).

Not all artificial datasets considered produced perfect results. In Figure 5b, we show an example of a dataset that includes ‘traffic jams’ that resemble more closely some of the phenomena in the actual sensor data. In many cases, traffic jams appear fairly rapidly, and then show an increased load on the bridge over a prolonged period. This is modeled in the data by medium-scale events that start and stop fairly rapidly, and remain constant in the meantime. The best result found, with cut-points $C = \{2^{12}, 2^{13}\}$, is shown in Figure 5b. This demonstrates that the proposed method is not able to properly separate the medium and low-scale events. In fact, even though the medium component does identify the

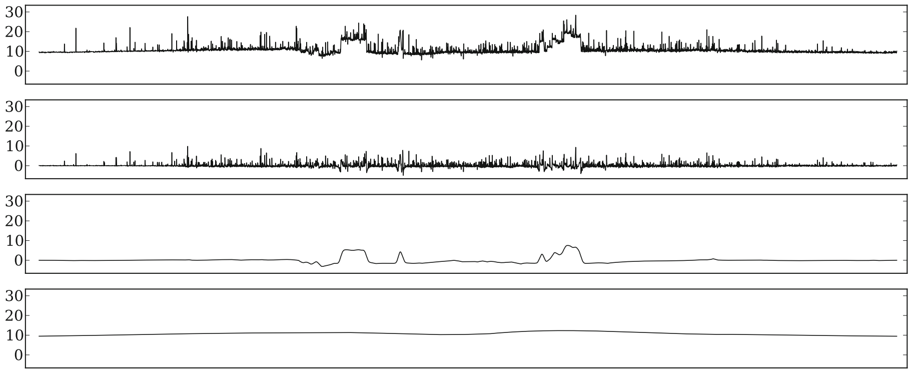


Fig. 6. Signal (top) and top-ranked scale decomposition for the InfraWatch data

location of the 'traffic jams', most of the rectangular nature is accounted for by the small scale. To some extent, this is understandable, as the start and end of the event could be considered high-frequency events with rapid changes in value. Therefore, parts of these events appear at a small scale, and the algorithm is mirroring this effect. In any case, the algorithm *is* able to identify the correct number of components, and is able to produce indications as to the location of the traffic jams. The top four results all show similar mixtures of scales, whereas the rank-five result groups the lowest two scales together. The $k = 1$ result appears at rank 14.

In order to better understand to what extent the proposed method is able to separate components at different scales, we carried out a more controlled experiment. We generated 11 different datasets constructed from 3 components. We fixed the scales of the short-term and long-term components respectively around $\sigma = 2^3$ and $\sigma = 2^{15}$, while the scale of the medium-term component varies from dataset to dataset in the range $(2^4, \ldots, 2^{14})$. The table below shows the number of components (k) of the top-ranked decomposition for the 11 datasets according to the scale parameter σ of the medium-term component.

σ	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
k	1	2	2	2	3	3	3	3	1	1	1

As the table suggests, the proposed method fails to identify the right number of components when the scales are too close to each other. However, when the scales are separated sufficiently ($2^8 \leq \sigma \leq 2^{11}$), the right number of components is identified. Also in this case, all the top-ranked decompositions relate to models that use polynomial representations.

InfraWatch Data. As anticipated by the motivating example in the introduction, we consider the strain measurements produced by a sensors attached to a large highway bridge in the Netherlands. For this purpose, we consider a time

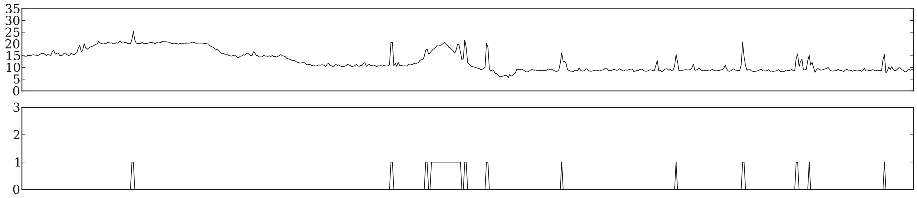


Fig. 7. A detail of the original strain signal (one hour) and the selected first component as represented with 4 symbols

series consisting of 24 hours of strain measurements sampled at 1 Hz (totaling $86,400$ data points). A plot of the data is shown in Figure 6 (topmost plot). We evaluated all the possible decompositions up to three components (two cut-points) allowing both the representation schemes we introduced. In the case of the discretization-based representations, we limit the possible cardinalities to 4, 16 and 64.

The top-ranked decomposition results in 3 components as shown in the last three plots in Figure 6. The selected cut-points appear at scales $2^6 = 64$ and $2^{11} = 2048$. All three components are represented with the discretization-based scheme, with a cardinality of respectively 4, 16, and 16 symbols. The decomposition has an MDL-score of $344,276$, where $L(M) = 19,457$ and $L(D \mid M) = 324,818$. The found components accurately correspond to physical events on the bridge. The first component, covering scales lower than 2^6, reflects the short-term influence caused by passing vehicles and represented as peaks in the signal. Note that the cardinality selected for this component is the lowest admissible in our setting (4). This is reasonable considering that the relatively simple dynamic behavior occurring at these scales, mostly the presence or not of a peak over a flat baseline, can be cheaply described with 4 or fewer states without incurring a too large error. The middle component, covering scales between 2^6 and 2^{11}, reflects the medium-term effects caused by traffic jams. As in the artificial data, the first component is slightly influenced by the second one, especially at the start and ending points of a traffic jam. Finally, the third component captures all the scales greater than 2^{11}, here representing the effect of temperature during a whole day. To sum up, the top-ranked decomposition successfully reflects the real physical phenomena affecting the data. The decompositions with rank 8 or less all present similar configurations of cut-points and cardinalities, resulting in comparable components where the conclusions above still hold. The first 2-component decomposition appears at rank 10 with the cut-point placed at scale 2^6, which separates the short-term peaks from all the rest of the signal (traffic jams and baseline mixed together). These facts make the result pretty stable as most of the good decompositions are ranked first.

An Application: Detecting Passing Vehicles. The component selection and representation generated by the MDL procedure may be useful in itself for tasks such as classification. For example, consider the short-term component of the previous example, Figure 6 (second plot). It represents the traffic activity over the bridge and has been represented with a discretization-based scheme using 4 symbols. Figure 7 shows a detail (1 hour) of the discretized component (bottom) and the relative original signal (top). The first 2 symbols (0 and 1) respectively classify the absence or presence of a passing vehicle, while the other two, considerably less frequent, are outliers in the data. The represented component, as selected by MDL, can thus be used to monitor traffic activity over the bridge, a task that is considerably more challenging using the original signal, due to the variations introduced by temperature fluctuations and traffic jams.

5 Related Work

Papadimitriou et al. [9] propose a method to discover the key trends in a time series at multiple time scales (window lengths) by defining an incremental version of Singular Value Decomposition. In signal processing, Independent Component Analysis [1] aims at separating a set of signals from a set of mixed signals but, in its standard formulation, requires at least as many sensors as sources. Our method is able to operate on a single input sensor and a variable number of sources to be discovered. Megalooikonomou et al. [8] introduce a multi-scale vector quantized representation of time series which enables fast and robust retrieval. The considered scales are however predefined and our approach could be used as a preprocessing step to determine those to include in the dictionary. The Minimum Description Length principle has been applied to the problem of choosing the best representation for a given time series by Hu et al. [3]. The authors propose a method to choose the best representation (and its parameters) among APCA, PLA and DFT. While there are similarities with our method (we also use the MDL principle to select the best model parameters for a given component), the authors put the stress on discovering the intrinsic cardinality of the data, other than its constituent multi-scale components. MDL has also been adopted to detect changes in the distribution of a data stream by van Leeuwen et al. [10].

6 Conclusions and Future Work

We introduced a novel methodology to discover the fundamental scale components in a time series in an unsupervised manner. The methodology is based on building candidate scale decompositions, defined over the scale-space image [12] of the original time series, with an MDL-based selection procedure aimed at choosing the optimal one.

A useful side product of the presented technique, due to the adoption of MDL, is that each discovered component is represented independently according to its inherent complexity and often results in a cheaper model (in terms of MDL score) in relation to the original raw time series. These cheaper per-component representations may better serve tasks like classification, regression or association analysis for time series produced by inherently multi-scale physical and artificial systems.

We have shown that our approach successfully identifies the relevant scale components in both artificial and real-world time series, giving meaningful insights about the data in the latter case. Future work will experiment with diverse representation schemes and hybrid approaches (such as using combinations of segmentation, discretization and Fourier-based encodings). Moreover, another interesting research question is how to substitute the presently employed exhaustive search of the optimal decomposition with a computationally cheaper heuristic approach, which is necessary in the case of large time series data.

Acknowledgments. This work and the InfraWatch project are funded by the Dutch funding agency STW, under project number 10970. It was partially funded by a Postdoc grant from the Research Foundation-Flanders.

References

1. Comon, P.: Independent component analysis, a new concept? Signal Processing 36(3), 287–314 (1994)
2. Grünwald, P.D.: The Minimum Description Length Principle. The MIT Press (2007)
3. Hu, B., Rakthanmanon, T., Hao, Y., Evans, S., Lonardi, S., Keogh, E.: Discovering the intrinsic cardinality and dimensionality of time series using mdl. In: Proceedings of ICDM 2011, pp. 1086–1091 (2011)
4. Keogh, E., Chu, S., Hart, D., Pazzani, M.: Segmenting time series: A survey and novel approach. In: Data mining in Time Series Databases, pp. 1–22 (1993)
5. Keogh, E.J., Chu, S., Hart, D., Pazzani, M.J.: An online algorithm for segmenting time series. In: Proceedings of ICDM 2001, pp. 289–296 (2001)
6. Knobbe, A., Blockeel, H., Koopman, A., Calders, T., Obladen, B., Bosma, C., Galenkamp, H., Koenders, E., Kok, J.: InfraWatch: Data Management of Large Systems for Monitoring Infrastructural Performance. In: Cohen, P.R., Adams, N.M., Berthold, M.R. (eds.) IDA 2010. LNCS, vol. 6065, pp. 91–102. Springer, Heidelberg (2010)
7. Lindeberg, T.: Scale-space for discrete signals. IEEE Trans. Pattern Analysis & Machine Intelligence 12(3), 234–254 (1990)
8. Megalooikonomou, V., Wang, Q., Li, G., Faloutsos, C.: A multiresolution symbolic representation of time series. In: Proceedings of ICDE 2005, pp. 668–679 (2005)
9. Papadimitriou, S., Yu, P.: Optimal multi-scale patterns in time series streams. In: Proceedings SIGMOD 2006, pp. 647–658. ACM (2006)
10. van Leeuwen, M., Siebes, A.: StreamKrimp: Detecting Change in Data Streams. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 672–687. Springer, Heidelberg (2008)
11. Vespier, U., Knobbe, A., Vanschoren, J., Miao, S., Koopman, A., Obladen, B., Bosma, C.: Traffic Events Modeling for Structural Health Monitoring. In: Gama, J., Bradley, E., Hollmén, J. (eds.) IDA 2011. LNCS, vol. 7014, pp. 376–387. Springer, Heidelberg (2011)
12. Witkin, A.P.: Scale-space filtering. In: Proceedings IJCAI 1983, San Francisco, CA, USA, pp. 1019–1022 (1983)

Separable Approximate Optimization of Support Vector Machines for Distributed Sensing

Sangkyun Lee, Marco Stolpe, and Katharina Morik

Fakultät für Informatik, LS VIII
Technische Universität Dortmund
44221 Dortmund, Germany
{sangkyun.lee,marco.stolpe,katharina.morik}@tu-dortmund.de

Abstract. Sensor measurements from diverse locations connected with possibly low bandwidth communication channels pose a challenge of resource-restricted distributed data analyses. In such settings it would be desirable to perform learning in each location as much as possible, without transferring all data to a central node. Applying the support vector machines (SVMs) with nonlinear kernels becomes nontrivial, however.

In this paper, we present an efficient optimization scheme for training SVMs over such sensor networks. Our framework performs optimization independently in each node, using only the local features stored in the respective node. We make use of multiple local kernels and explicit approximations to the feature mappings induced by them. Together they allow us constructing a separable surrogate objective that provides an upper bound of the primal SVM objective. A central coordination is also designed to adjust the weights among local kernels for improved prediction, while minimizing communication cost.

Keywords: distributed features, support vector machines, separable optimization, primal formulation, approximate feature mappings.

1 Introduction

Sensor networks have been a very active research topic in recent machine learning and data mining [12,13]. Sensors are adopted to monitor certain aspects of objects or phenomena that we are interested in, often located in such places hardly accessible by human beings. Various applications include monitoring manufacturing processes, traffic levels, water flows and climate changes over time at different locations, where sensors (or computing nodes embracing local sensors) have to communicate with each other or with a central arbitrator in order to provide useful information for decision making.

Challenges arise in sensor networks when we try to build a predictor collecting information from all sensors, where sensors can afford only minimal communication due to their distance to a central station or low-power requirements. If this is the case, we might prefer to perform learning in a distributed fashion, where each separated part of learning relies on locally stored measurements only. Learning a global model in such cases requires an approach whose computation can

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 387–402, 2012.

be distributed in a well-defined way, equipped with a global arbitration that can maximize prediction performance without incurring too much information transfer from sensors.

In this paper we suggest a variant of the support vector machines (SVMs) using nonlinear kernels on sensor data. We define kernels that use only locally stored features in sensors, computing explicit forms of approximations to the feature mappings that correspond to each local kernel. For typical error functions of SVMs, we then create a separable surrogate objective function that forms an upper bound on the original primal SVM objective. Each separated part in the objective is designed to use a single local kernel, and therefore can be optimized locally at the sensors.

We also provide an additional central optimization that uses inner product results from the sensors, without requiring an access to local kernel functions or their approximations. The central optimization provides local kernels new weights, that can be fed to the sensors and generate possibly improved local solutions.

2 Separable Optimization

In this section we begin with a general description of the support vector machines (SVMs), shaping it progressively to a form which can be optimized separately for local features in each network node.

2.1 Support Vector Machines

We consider a given data set $\{(\mathbf{x}_u, y_u)\}_{u=1}^m$ which consists of pairs of input feature vectors $\mathbf{x}_u \in \mathbb{R}^p$ and their labels y_u, where $y_u \in \{-1, +1\}$ for classification and $y_u \in \mathbb{R}$ for regression. The SVMs for 1- and 2-class classification and regression can be formulated as an unconstrained convex minimization,

$$\min_{\mathbf{w}\in\mathcal{H},\rho\in\mathbb{R}} \ \frac{\lambda}{2}\left(\|\mathbf{w}\|_{\mathcal{H}}^2 + \rho\right) + \frac{1}{m}\sum_{u=1}^{m} \ell\left(\langle \mathbf{w}, \phi(\mathbf{x}_u)\rangle, y_u, \rho\right) \tag{1}$$

where $\lambda > 0$ and ℓ is a convex loss function chosen by the task of interest as in Table 1. For readability we ignore the intercept term of a decision plane without loss of generality, which can be easily included by augmenting vectors $\mathbf{w}$ and $\mathbf{x}$ and excluding it from penalization in the first objective term. We call $\phi : \mathbb{R}^p \to \mathcal{H}$, for a Hilbert space $\mathcal{H}$, a *feature mapping* induced by a positive semidefinite kernel k, with the relation that $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}')\rangle$ for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$.

2.2 Multiple Localized Kernels

Now, we consider that the input features are stored in a distributed fashion among n nodes, possibly with overlaps among them. We suppose that there are p unique features in total, denoting by $\mathcal{S}_i \subset \{1, 2, \ldots, p\}$ the subset of feature

Table 1. The loss function ℓ and the range of ρ in the canonical objective of SVMs in the equation (1), and ℓ_i and ρ_i corresponding to the ith summand in the upper bounds of ℓ formed in (6). For the training example indexed by u, we set $z = \langle \mathbf{w}, \phi(\mathbf{x}_u)\rangle$ and $z_i = \mathbf{w}[i]^T \varphi_i(\mathbf{x}_u[i])$. The zero range for ρ and ρ_i implies we ignore them in optimization.

Task	$\ell(z, y, \rho)$	$\ell_i(z_i, y, \rho_i)$	Range ρ, ρ_i
Classification (1-class)	$\max\{0, \rho - z\}$	$\max\{0, \rho_i - z_i\}$	$\mathbb{R}$
Classification (2-class)	$\max\{0, 1 - yz\}$	$\max\{0, 1 - yz_i\}$	0
Regression	$\max\{0, \lvert y - z\rvert - \epsilon\}$	$\max\{0, \lvert y - z_i\rvert - \epsilon\}$	0

indices stored in the ith node, and by $p_i := |\mathcal{S}_i| > 0$ its cardinality, so that $\cup_{i=1}^{n} \mathcal{S}_i = \{1, 2, \ldots, p\}$ and $\sum_{i=1}^{n} p_i \geq p$. For convenience, we refer to the feature subvector of $\mathbf{x}_u$ stored in the ith node as $\mathbf{x}_u[i] \in \mathbb{R}^{p_i}$.

For each node we make use of an individual kernel which depends on only the features stored locally in nodes. We denote the kernel for the ith node by $k_i : \mathbb{R}^{p_i \times p_i} \to \mathbb{R}$ and its corresponding feature mapping by $\phi_i : \mathbb{R}^{p_i} \to \mathcal{H}_i$. Then we can construct a *composite* kernel k as a conic combination of local kernels, that is,

$$k(\mathbf{x}, \mathbf{x}') := \sum_{i=1}^{n} \mu_i^2 k_i(\mathbf{x}[i], \mathbf{x}'[i]). \tag{2}$$

(It will become clear why we use μ_i^2 rather than $\mu_i \geq 0$, as we progress.) The weights for local kernels μ_i will be optimized, which defines our central optimization problem to be discussed later. This setting is very similar to the multiple kernel learning (MKL) and boosting, but our resulting framework will not be exactly the same, as we discuss later in Section 3.

We note that using multiple kernels alone does not lead to a separable objective for SVMs. From the representer theorem [20], the optimal weight $\mathbf{w}$ of SVM (1) is expressed as a linear span of the representers $k(\cdot, \mathbf{x}_v)$. That is,

$$\mathbf{w}(\cdot) = \sum_{v=1}^{m} \alpha_v k(\cdot, \mathbf{x}_v) \overset{(2)}{=} \sum_{i=1}^{n} \mu_i^2 \sum_{v=1}^{m} \alpha_v k_i(\cdot[i], \mathbf{x}_v[i]), \tag{3}$$

Replacing $\mathbf{w}$ into (1) results in the following objective:

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^m} \; & \frac{\lambda}{2} \sum_{i=1}^{n} \mu_i^2 \sum_{u=1}^{m} \sum_{v=1}^{m} \alpha_u \alpha_v k_i(\mathbf{x}_u[i], \mathbf{x}_v[i]) \\ & + \frac{1}{m} \sum_{u=1}^{m} \ell \left(\sum_{i=1}^{n} \mu_i^2 \sum_{v=1}^{m} \alpha_v k_i(\mathbf{x}_u[i], \mathbf{x}_v[i]), y_u, \rho \right). \end{aligned}$$

We can see that all optimization variables $\alpha_1, \alpha_2, \ldots, \alpha_m$ are coupled with each node $i = 1, 2, \ldots, n$, therefore cannot be split over nodes. This also indicates that we would need alternative ways to incorporate kernels rather than relying on the representer theorem, to achieve separability.

2.3 Approximating Feature Mappings

One observation of the original SVM formulation (1) is that the terms $\|\mathbf{w}\|^2$ and $\langle \mathbf{w}, \phi(\mathbf{x}_u)\rangle$ will be separable over the components of $\mathbf{w}$, if $\mathbf{w}$ and $\phi(\mathbf{x}_u)$ are in a finite dimensional space. (The loss functions ℓ in Table 1 are not separable as well. We will discuss them as the next step in the following section.) Motivated by this, we introduce explicit finite-dimensional approximations to the feature mappings $\phi \in \mathcal{H}$ that correspond to kernel functions $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}')\rangle$. This step is necessary, since the explicit form of ϕ is unavailable in general.

For each node i, suppose that we have obtained an approximate feature mapping $\varphi_i : \mathbb{R}^{p_i} \to \mathbb{R}^{d_i}$ to the original mapping ϕ_i, where $d_i \in (0, \infty)$ is a predefined (possibly small) integer, so that

$$\langle \varphi_i(\mathbf{x}[i]), \varphi_i(\mathbf{x}'[i])\rangle \approx k_i(\mathbf{x}[i], \mathbf{x}'[i]) = \langle \phi(\mathbf{x}[i]), \phi(\mathbf{x}'[i])\rangle .$$

When $d := \sum_{i=1}^n d_i$ is sufficiently large, we can consider the following d-dimensional problem as a good approximation to the original problem (1):

$$\min_{\mathbf{w}\in\mathbb{R}^d, \rho\in R} \frac{\lambda}{2}\left(\|\mathbf{w}\|_2^2 + \rho\right) + \frac{1}{m}\sum_{u=1}^m \ell\left(\mathbf{w}^T\varphi(\mathbf{x}_u), y_u, \rho\right) . \tag{4}$$

Regarding the representation (3), we impose a weight $\mu_i \geq 0$ for the feature mappings in nodes $i = 1, 2, \ldots, n$, so that

$$\varphi(\mathbf{x}) := \begin{bmatrix} \mu_1\varphi_1(\mathbf{x}[1]) \\ \mu_2\varphi_2(\mathbf{x}[2]) \\ \vdots \\ \mu_n\varphi_n(\mathbf{x}[n]) \end{bmatrix}, \ \mathbf{w} := \begin{bmatrix} \mathbf{w}[1] \\ \mathbf{w}[2] \\ \vdots \\ \mathbf{w}[n] \end{bmatrix} \ \Rightarrow \ \mathbf{w}^T\varphi(\mathbf{x}) = \sum_{i=1}^n \mu_i \mathbf{w}[i]^T\varphi_i(\mathbf{x}[i]).$$

Here we denote by $\mathbf{w}[i] \in \mathbb{R}^{d_i}$ the subvector of $\mathbf{w} \in \mathbb{R}^d$ for the node i. Again, by $\mathbf{x}[i] \in \mathbb{R}^{p_i}$ we denote the attributes of $\mathbf{x} \in \mathbb{R}^p$ stored in the node i.

In the expansion of $\mathbf{w}^T\varphi(\mathbf{x})$ above, we have μ_i but no μ_i^2 as in (2) or (3). The reason is that we do not have any inner product between images of $\varphi_i(\cdot)$: this becomes a crucial property for deriving a separable optimization problem.

There have been largely two types of approaches to find approximate feature mappings φ. The first type of approaches is based on computing low-rank factor matrices that approximate the original kernel matrices [4,3,6]. Although this type allows to use any positive semidefinite kernel matrices, it requires matrix factorization with comparably large memory footprint.

In the second type, we make use of random projections and construct approximate mappings directly [16]. These methods tend to require larger approximation dimensions than the first type [11], but they are much simpler and easier to parallelize. In this paper we focus on the second type approximation of the Gaussian kernels $k_i(\mathbf{x}[i], \mathbf{x}'[i]) = \exp(-\gamma_i\|\mathbf{x}[i] - \mathbf{x}'[i]\|_2^2)$ for some $\gamma_i > 0$ without loss of generality, for which the approximation is given by

$$\varphi_i(\mathbf{x}[i]) = \sqrt{\frac{2}{d_i}}\left[\cos(\mathbf{z}_1^T\mathbf{x}[i] + e_1), \cos(\mathbf{z}_2^T\mathbf{x}[i] + e_2), \ldots, \cos(\mathbf{z}_{d_i}^T\mathbf{x}[i] + e_{d_i})\right]^T , \tag{5}$$

for each node i, where $\mathbf{z}_j \in \mathbb{R}^{p_i}$ and $e_j \in \mathbb{R}$ are i.i.d. random samples from the Gaussian distribution $\mathcal{N}(\mathbf{0}, 2\gamma_i I)$, I is an identity matrix, and from the uniform distribution on $[0, 2\pi]$, respectively. These are derived from the Fourier transform of the kernel function k_i (for more details see [16]). Note that φ_i uses only local features stored in the ith node, represented as a subvector $\mathbf{x}[i]$.

2.4 A Separable Surrogate Objective

Our final step is to make the loss functions ℓ in Table 1 separable over nodes, using their convexity in the first and the last arguments. For this purpose we impose $\sum_{i=1}^n \mu_i = 1$ in addition to $\mu_i \geq 0$. Then we can derive an upper bound for the hinge loss ℓ of 2-class classification as follows,

$$\begin{aligned}\ell(\mathbf{w}^T\varphi(\mathbf{x}), y, \rho) &= \max\{0, 1 - y\mathbf{w}^T\varphi(\mathbf{x})\} \\ &= \max\{0, \sum_{i=1}^n \mu_i(1 - y\mathbf{w}[i]^T\varphi_i(\mathbf{x}[i]))\} \\ &\leq \sum_{i=1}^n \mu_i \ell_i\left(\mathbf{w}[i]^T\varphi_i(\mathbf{x}[i]), y, \rho_i\right) \qquad (6)\end{aligned}$$

where ℓ_i is listed in the second row of Table 1, and we define ρ_i so that $\rho = \sum_{i=1}^n \mu_i\rho_i$. The upper bounds for 1-class classification and regression tasks can be derived similarly and are presented in the table as well. Summing up the inequalities (6) over training indices $u = 1, 2, \ldots, m$ leads to an upper bound of the objective function in (4):

$$\begin{aligned}&\frac{\lambda}{2}\left(\|\mathbf{w}\|_2^2 + \rho\right) + \frac{1}{m}\sum_{u=1}^m \ell\left(\mathbf{w}^T\varphi(\mathbf{x}_u), y_u, \rho\right) \\ &\leq \sum_{i=1}^n \left[\frac{\lambda}{2}\left(\|\mathbf{w}[i]\|_2^2 + \mu_i\rho_i\right) + \frac{1}{m}\sum_{u=1}^m \mu_i\ell_i\left(\mathbf{w}[i]^T\varphi_i(\mathbf{x}_u[i]), y_u, \rho_i\right)\right]. \qquad (7)\end{aligned}$$

The expression in the right hand side is separable in terms of nodes. Therefore, in each node $i = 1, 2, \ldots, n$ we can solve the following separated problem,

$$\textbf{(Local)} \quad \min_{\mathbf{w}[i]\in\mathbb{R}^{d_i}, \rho_i\in\mathbb{R}} \frac{\lambda}{2}\left(\|\mathbf{w}[i]\|_2^2 + \mu_i\rho_i\right) + \frac{1}{m}\sum_{u=1}^m \mu_i\ell_i\left(\mathbf{w}[i]^T\varphi_i(\mathbf{x}_u[i]), y_u, \rho_i\right). \qquad (8)$$

Although it is possible to construct a global classifier by transferring all local solutions $\mathbf{w}^*[i]$ and ρ_i^* of (8) to a central node for $i = 1, 2, \ldots, n$, it may not be desirable since then the central node should know about φ_i and local features as well. This requires $\mathcal{O}(\sum_{i=1}^n d_i p_i)$ numbers to be transferred, plus $\mathcal{O}(\sum_{i=1}^n p_i)$ per test point $\mathbf{x}$ whose features are stored in a distributed fashion. Instead, we let each node compute and transfer two scalars $\mathbf{w}^*[i]^T\varphi_i(\mathbf{x}[i])$ and ρ_i^* to a central node, where weighted summations $\mathbf{w}^{*T}\varphi(\mathbf{x}) = \sum_{i=1}^n \mu_i\mathbf{w}^*[i]^T\varphi_i(\mathbf{x}[i])$ and $\rho^* = \sum_{i=1}^n \mu_i\rho_i^*$ can be used for global prediction. This approach reduces the communication cost to $\mathcal{O}(n)$ for a test point.

2.5 Minimization of Approximation Gaps

We discuss the quality of two approximations we have made, in using approximate feature mappings and an inequality due to the convexity of loss functions, to arrive the separated local optimization (8) from the nonseparable SVM formulation (1), assuming that both are using multiple localized kernels.

Approximation in Feature Mappings. The first approximation has been applied when we use approximate feature mappings in Section 2.3. In the case of the mapping $\varphi_i : \mathbb{R}^{p_i} \to \mathbb{R}^{d_i}$ in (5) approximating a local Gaussian kernel $k_i(\mathbf{x}[i], \mathbf{x}'[i]) = \exp(-\gamma_i \|\mathbf{x}[i] - \mathbf{x}'[i]\|_2^2)$, $\mathbf{x}[i] \in \mathbb{R}^{p_i}$, the following result from [16] quantifies its quality:

$$\mathbb{P}\left[\sup_{\mathbf{x}[i],\mathbf{x}'[i]\in\mathcal{M}} \left|\varphi_i(\mathbf{x}[i])^T\varphi_i(\mathbf{x}'[i]) - k_i(\mathbf{x}[i],\mathbf{x}'[i])\right| \geq \epsilon\right] \leq \mathcal{O}\left(\epsilon^{-2}e^{-\frac{\epsilon^2 d_i}{4(p_i+2)}}\right),$$

where $\mathcal{M} \subset \mathbb{R}^{p_i}$ is a compact set containing all subvectors $\mathbf{x}_u[i]$, $u = 1, 2, \ldots, m$. Therefore, φ_i grants us good approximation as long as we use sufficiently large d_i for its approximation dimension.

Approximation in the Convex Inequality. Another approximation takes place in (6) and (7), where we construct separable upper bounds of the nonseparable loss functions ℓ in Table 1. Since the inequality is constructed using the convex combination parametrized by $\mu_1, \mu_2, \ldots, \mu_n$, we can reduce the gap in the inequality by minimizing the right hand side expression of (7) in terms of μ_i's. This defines an optimization problem in a central node,

$$\begin{aligned} & \min_{\mu:=(\mu_1,\mu_2,\ldots,\mu_n)^T} \frac{1}{m}\sum_{u=1}^{m}\sum_{i=1}^{n} L_{ui}\mu_i + \Psi(\mu) \\ \textbf{(Central)}\quad & \text{s.t.} \sum_{i=1}^{n} L_{ui}\mu_i \geq \ell\left(\sum_{i=1}^{n} Z_{ui}\mu_i, y_u, \sum_{i=1}^{n}\mu_i\rho_i\right),\ u = 1, 2, \ldots, m, \\ & \sum_{i=1}^{n}\mu_i = 1,\ \ \mu_i \geq 0,\ i = 1, 2, \ldots, n. \end{aligned} \tag{9}$$

Here we have defined

$$\begin{cases} Z_{ui} & := \mathbf{w}[i]^T\varphi_i(\mathbf{x}_u[i]) \\ L_{ui} & := \ell_i(Z_{ui}, y_u, \rho_i) \end{cases},\quad u = 1, 2, \ldots, m,\ i = 1, 2, \ldots, n. \tag{10}$$

The constants in Z_{ui} can be computed independently in local nodes and transferred to the central node.

The last term Ψ in the objective of (9) is an optional convex regularization term. This can be chosen as $\Psi(\mu) = \frac{\sigma}{2}\|\mu\|_2^2$ for some $\sigma > 0$ to produce a unique or an evenly distributed solution, $\Psi(\mu) = \sigma'\|\mu\|_1$ to induce elementwise sparsity in μ (thereby selecting few local kernels important for prediction, similarly to MKL), or $\Psi(\mu) = \sigma'' \sum_{g=1}^{G} \|\mu[g]\|_2$ for subvectors $\mu[g]$ to promote groupwise sparsity (e.g. selecting few clusters of nodes, rather than individual nodes).

3 Related Works

We present two most closely related learning approaches to our framework.

3.1 Multiple Kernel Learning

The multiple kernel learning (MKL) is an extension of the support vector machines for employing multiple kernel functions, instead of a single one as in the standard settings. The current forms and efficient learning methods for MKL have been established in [10,9,1,2]. In MKL we consider a combination of n kernels

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} \beta_i k_i(\mathbf{x}, \mathbf{x}'), \;\; \beta_i \geq 0, \;\; \sum_{i=1}^{n} \beta_i = 1, \tag{11}$$

and k_i's are defined on a certain subset of features. Plugging the composite kernel $k(\mathbf{x}, \mathbf{x}')$ into the standard SVM formulation leads to a semi-definite program (SDP) [10], which is much harder to solve than the standard SVMs. When kernels k_i are normalized, i.e. $k_i(\mathbf{x}, \mathbf{x}) = 1$, it can be reduced to a quadratically constrained quadratic program [9], which can be solved slightly more efficiently than SDPs. Modifications to the SVM formulations lead to further improvement, resulting in a semi-infinite linear program [19], a quadratic program [17], or a much faster interleaved optimization using ℓ_p-norms [8].

The main difference of MKL to our framework is that the objective function of MKL is not separable over nodes. For instance, the MKL formulation in [17] solves the dual problem for fixed weights $\beta_1, \beta_2, \ldots, \beta_n$,

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^m} \quad & -\frac{1}{2} \sum_{i=1}^{n} \beta_i \sum_{u=1}^{m} \sum_{v=1}^{m} \alpha_u \alpha_v k_i(\mathbf{x}_u, \mathbf{x}_v) + \sum_{u=1}^{m} \alpha_u \\ \text{s.t.} \quad & \sum_{u=1}^{m} \alpha_u y_u = 0, \;\; 0 \leq \alpha_u \leq 1/(m\lambda), \;\; u = 1, 2, \ldots, m. \end{aligned}$$

Similar to our discussion in Section 2.2, all variables α_u's in this objective are coupled with each node i, therefore the optimization cannot be separated over nodes. Another difference is the ways to form the convex combinations of kernels. Comparing the convex combinations in (11) and (2), we can interpret our μ_i as $\sqrt{\beta_i}$, and we impose $\sum_{i=1}^{n} \mu_i = 1$, rather than $\sum_{i=1}^{n} \beta_i = 1$.

3.2 Boosting

Boosting with an additive model [5] is also quite similar to our model and MKL. In boosting, we find a linear combination of n basis functions or weak learners of the form

$$h(\mathbf{x}) = \sum_{i=1}^{n} \zeta_i h_i(\mathbf{w}; \mathbf{x}),$$

where $\zeta_i \in \mathbb{R}$, and h_i can be set $h_i(\mathbf{w}; \mathbf{x}) = \mathbf{w}[i]^T \varphi_i(\mathbf{x}[i])$ to make it similar to our setting. The optimal $\mathbf{w}^*[i]$ and ρ_i^* can be found independently for each node $i = 1, 2, \dots, n$, and the best combination of h_i's can be found by solving

$$\min_{\zeta_1, \dots, \zeta_n} \sum_{u=1}^{m} \ell \left(\sum_{i=1}^{n} \zeta_i h_i(\mathbf{w}^*; \mathbf{x}_u), y_u, \sum_{i=1}^{n} \zeta_i \rho_i^* \right).$$

This resembles our central problem (9). Despite its similarity, however, the objective here does not provide an upper bound of the MKL objective as in our formulation (7), thereby losing its connection to MKL. Also, unlike our setting and MKL, the local problems in boosting do not depend on the weights ζ_i. That is, the solutions from separated problems cannot be improved any further using updated weight values of ζ_i obtained from the central optimization. Our framework and MKL share the property that subproblems (separated problems in our case, and the nonseparable problem obtained after fixing kernel weights in MKL) are dependent on such weights, therefore we can obtain improved solutions using adjusted weight values.

4 Algorithm

We describe our algorithm that solves the separated problem (8) at each local node, and an additional central optimization (9) that determines the optimal convex combination. The outline of the entire framework is presented in Algorithm 1.

4.1 Local Optimization

To find the solutions of each separated local optimization problem, we use the stochastic gradient descent (SGD) approach. In particular, we adapt the "robust" version of SGD suggested by Nemirovski and Yudin [15], for which a simplified analysis [14] or a regret-based online learning analysis [21] can be found. We sketch the robust SGD algorithm here and refer to the ASSET approach [11] for details, which implements essentially the same idea for the standard SVMs.

To simplify our discussion, we denote the objective function of the ith separated local problem in (8) by f_i:

$$f_i(\mathbf{w}[i], \rho_i) := \frac{\lambda}{2} \left(\|\mathbf{w}[i]\|_2^2 + \mu_i \rho_i \right) + \frac{1}{m} \sum_{u=1}^{m} \mu_i \ell_i \left(\mathbf{w}[i]^T \varphi_i(\mathbf{x}_u[i]), y_u, \rho_i \right).$$

Then in each iteration of the local optimization, we update the variables $\mathbf{w}[i]$ and ρ_i as follows,

$$\begin{bmatrix} \mathbf{w}[i]^{t+1} \\ \rho_i^{t+1} \end{bmatrix} \leftarrow \mathcal{P}_{\mathcal{W}} \left(\begin{bmatrix} \mathbf{w}[i]^{t} \\ \rho_i^{t} \end{bmatrix} - \eta_t G_t \right), \quad t = 1, 2, \dots, T, \qquad (12)$$

Algorithm 1. Separable SVM with Approximations to Local Kernels

```
input : A data set $\{(\mathbf{x}_u, y_u)\}_{u=1}^m$, the number of iterations $K$ ($K > 1$ only if
        we use central optimization), positive integers $T$ and $T_0$, and $\mu_0 = 1/n$.
Initialize: $\mu_i \leftarrow \mu_0$, for $i = 1, 2, \ldots, n$;
for $k = 1, 2, \ldots, K$ do
    Transmit $\mu_i$ to all nodes $i = 1, 2, \ldots, n$;
    (local: in parallel)
        // Solve a separated local problem in each node $i$, using ASSET
        input : a weight $\mu_i$ and local measurements/labels $\{(\mathbf{x}_u[i], y_u)\}_{u=1}^m$.
        Initialize iterates and averages: $\mathbf{w}[i]^1 \leftarrow \mathbf{0}$, $\rho_i^1 \leftarrow 0$ $\bar{\mathbf{w}}[i] \leftarrow \mathbf{0}$, $\bar{\rho}_i \leftarrow 0$;
        Estimate an optimization constant $\theta_i > 0$;
        for $t = 1, 2, \ldots, T$ do
            Select a random training index $\xi_t \in \{1, 2, \ldots, m\}$;
            Compute a steplength $\eta_t = \theta_i/\sqrt{t}$;
            Update $\mathbf{w}[i]$ and $\rho_i$ via (12);
        end
        output: averages of iterates, $\bar{\mathbf{w}}[i]$ and $\bar{\rho}_i$, for the last $(T - T_0)$ iterations.
        output: (optional) transfer $Z_{ui} := \bar{\mathbf{w}}[i]^T \varphi_i(\mathbf{x}_u)$ for $u = 1, 2, \ldots, m$ and
                $\bar{\rho}_i$ to a central node.
    (end)
    (central: optional)
        // Solve a central problem, using CPLEX
        input : $Z_{ui}$ and $\bar{\rho}_i$ for $u = 1, 2, \ldots, m$, $i = 1, 2, \ldots, n$.
        Compute $L_{ui} := \ell_i (Z_{ui}, y_u, \bar{\rho}_i)$ for all $u = 1, 2, \ldots, m$, $i = 1, 2, \ldots, n$;
        Compute $\rho = \sum_{i=1}^n \mu_i \bar{\rho}_i$;
        Solve an equivalent formulation (13);
        output: new weights $\mu_1, \mu_2, \ldots, \mu_n$.
    (end)
end
```

where $\mathcal{P}_{\mathcal{W}}(\mathbf{z}) := \arg\min_{\mathbf{v}\in\mathcal{W}}\{\frac{1}{2}\|\mathbf{z} - \mathbf{v}\|_2^2\}$ is an Euclidean projection of a vector $\mathbf{z}$ onto a convex set $\mathcal{W}$, G_t is an estimate subgradient of f_i at $(\mathbf{w}[i]^t, \rho_i^t)$, constructed using a training example chosen by a random index $\xi_t \in \{1, 2, \ldots, m\}$, and η_t is a steplength of the form $\eta_t = \theta_i/\sqrt{t}$ for some $\theta_i > 0$. The set $\mathcal{W}$ guides the optimization to avoid taking too large steps, whose formulation can derived analytically from strong duality [18,11].

The convergence of the robust SGD algorithm is $\mathcal{O}(c(T_0/T)\theta_i/\sqrt{T})$ in terms of objective function values in expectation, where $c(\cdot)$ is a simple function only depending on the ratio T_0/T [14].

4.2 Central Optimization

The central problem (9), for the loss functions ℓ in Table 1, can be formulated as a linear program (LP) or a quadratic program (QP) depending on the choices of the regularizer Ψ. When we consider 2-class problems with $\Psi(\mu) = \frac{\sigma}{2}\|\mu\|_2^2$ for $\sigma \geq 0$, we can write an equivalent formulation to (9) as follows,

$$\begin{aligned} \min_{\mu \in \mathbb{R}^n} \ & \frac{1}{m}\mathbf{1}_m^T L\mu + \frac{\sigma}{2}\mu^T\mu, \\ \text{s.t.} \ & (L + D_y Z)\mu \geq \mathbf{1}_m, \ \ \mathbf{1}_n^T\mu = 1, \ \ \mu \geq \mathbf{0}, \end{aligned} \tag{13}$$

where the elements of the matrices $L \in \mathbb{R}^{m\times n}$ and $Z \in \mathbb{R}^{m\times n}$ are defined in (10), D_y is a diagonal matrix with elements $y_1, y_2, \ldots, y_m$, and $\mathbf{1}_m := (1, 1, \ldots, 1)^T \in \mathbb{R}^m$. Similar formulations can be derived for 1-class and regression tasks.

The solutions of (13) can be obtained using LP solvers when $\sigma = 0$, or using QP solvers for $\sigma > 0$. In our experiments we use $\sigma = 0.5$, since it has produced slightly better solutions than using $\sigma = 0$. For the solution method we adopt the IBM ILOG CPLEX Optimization Studio Academic Research Edition v.12.4, which provides one of the fastest LP/QP solvers for free for academic institutes.

The total number of elements to be transferred to a central node is $\mathcal{O}(m)$ for each node $i = 1, 2, \ldots, n$. (These elements compose the matrix Z.) This cost can be reduced, trading some potential prediction improvement, by transferring information for a small subsample of size $m' < m$, rather than for the entire training set of size m. This also reduces the number of constraints in the central problem (13) from $\mathcal{O}(m)$ to $\mathcal{O}(m')$. We have used $m' = 5000$ for our experiments.

We set the maximum number of central optimization to $K = 10$, stopping the algorithm earlier if the prediction performance on m' training samples does not improve any further. (Three passes were enough in most cases.)

5 Experiments

We implemented Algorithm 1 based on the open-source `C++` program ASSET [11][1], comparing several different implementations built upon it:

- `Separated`: implements Algorithm 1.
- `Composite`: the standard SVM with a composite kernel (2) consisting of local kernels. We set $\mu_i = \frac{1}{n}$ for all $i = 1, 2, \ldots, n$.
- `Single`: the standard SVM with a single global kernel that uses all features.

All of these make use of approximations to the kernel feature mappings. We also use `SVMLight` with its default parameters to make comparisons to the cases using exact kernel information.

In all runs, we randomly partition the set of features into equal-sized n groups and assign each group to one of n nodes. We use an *overlap* parameter to specify the percentage of features in each node that are sampled from other nodes, simulating peer-to-peer information exchange among nodes. The purpose of such communication will be amending the loss of information due to partitioning.

We use Gaussian kernels of the form $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|_2^2)$ in all experiments, where the parameter γ is tuned by a cross validation using `SVMLight` [7]

[1] Available at `http://pages.cs.wisc.edu/~sklee/asset/`

Table 2. Data sets and their training parameters

Name	m (train)	test	p	(density)	λ	γ
ADULT	40701	8141	124	(11.2%)	3.07e-08	0.001
MNIST	58100	11900	784	(19.1%)	1.72e-07	0.01
CCAT	89702	11574	47237	(1.6%)	1.28e-06	1.0
IJCNN	113352	28339	22	(56.5%)	8.82e-08	1.0
COVTYPE	464809	116203	54	(21.7%)	7.17e-07	1.0

for real-world data sets. The parameter λ is tuned in the same way, and both are shown in Table 2. For artificial data we use $\lambda = 0.133$ and $\gamma = 0.001$ found by the `Single` code. Whenever we have localized kernels $k_i = \exp(-\gamma_i \|\mathbf{x}[i] - \mathbf{x}'[i]\|_2^2)$, we set their parameters by $\gamma_i = n\gamma$. The purpose here is to compensate the difference between the orders of magnitude $\|\mathbf{x}[i] - \mathbf{x}'[i]\|_2^2 \in \mathcal{O}(p_i)$ and $\|\mathbf{x} - \mathbf{x}'\|_2^2 \in \mathcal{O}(p)$, in a way that makes the arguments for exponential functions similar, i.e.

$$\gamma_i = \frac{p}{p_i}\gamma \;\approx\; n\gamma \;\Rightarrow\; \gamma_i \|\mathbf{x}[i] - \mathbf{x}'[i]\|_2^2 \in \mathcal{O}(\gamma p).$$

For creating approximate feature mappings for kernels, we set their dimensions to $d = 1000$ and $d_i \approx d/n$ for all experiments.

All experiments have been performed on 64-bit multicore Linux systems, where a thread is created to simulate a node optimizing a separated objective.

5.1 Data

We use an artificial data set and five real-world benchmark data sets.

Artificial Data. A data set is created by sampling 7500 (training) and 2500 (testing) $p = 64$ dimensional random vectors from two multivariate Gaussian distributions, $\mathcal{N}_-(\eta_-, \Sigma)$ and $\mathcal{N}_+(\eta_+, \Sigma)$ for two classes. We fix $\eta_- = (-1, \ldots, -1)^T$ and $\eta_+ = (1, \ldots, 1)^T$. The two distributions share a covariance matrix Σ, which is constructed with controlling the maximum number of nonzero entries (the ratio is specified by the *correlation ratio* $r \in [0, 1]$). To construct a positive semidefinite matrix Σ, we first sample a random matrix $S \in \mathbb{R}^{p \times p}$, computing its QR decomposition, $S = QR$. Then we replace a fraction r of the rows of Q by normalized random vectors of length p. Finally we set $\Sigma = QQ^T$, so that Σ will contain up to $p(rp + (1 - r)(rp + 1))$ nonzero entries.

Real-World Data Sets. Five real-world benchmark data sets[2] in Table 2 are prepared as follows. `ADULT` is from the UCI machine learning repository, randomly split into training and test sets. `MNIST` is prepared for classifying the digits 0-4 from 5-9. `CCAT` is from the RCV1 collection, classifying the category

[2] Available at the UCI Repository `http://archive.ics.uci.edu/ml/`, or at the LIBSVM website `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

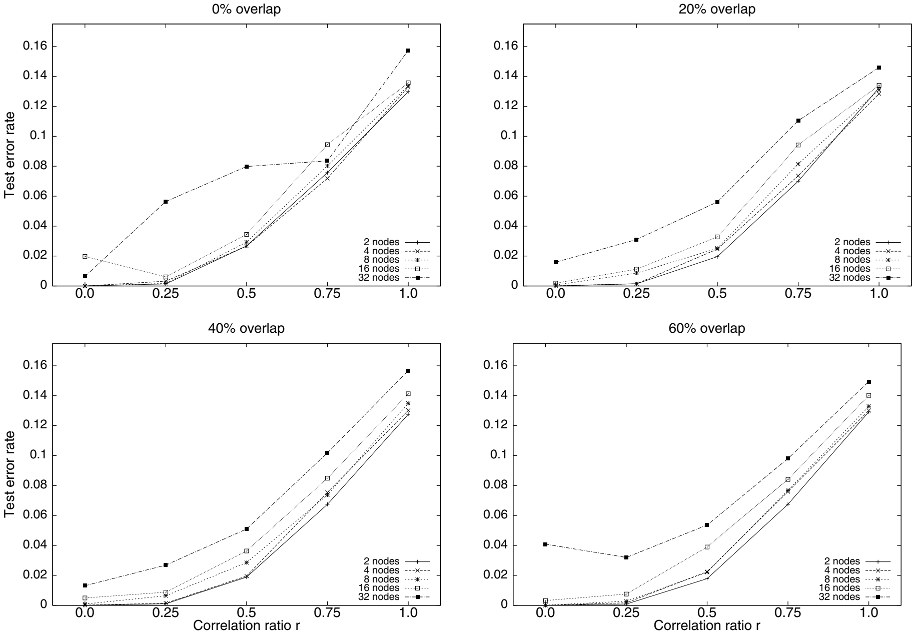


Fig. 1. Test error rates for different overlaps of features (100%: all features are available at each local node), numbers of nodes, and correlation ratios (1.0: nonzero correlation between all pairs of features). All measurements are averages over 20 runs with random splits of features and random approximations of φ.

CCAT from the others, where we use the original test set as our training set and the original training set as our test set. IJCNN is from the IJCNN 2001 Challenge data set. COVTYPE classifies type 1 against the other forest cover types.

5.2 Artificial Data

Locality of Information vs. Prediction Performance. We first evaluated how the locality of features affects the prediction performance of our algorithm using localized kernels.

In Figure 1, we show the test error rates for different overlaps of attributes, numbers of nodes, and correlation ratio r (averaged values over 20 repetitions using randomized splits of features and approximations of φ). In each plot, for a fixed number of nodes, say $n = 8$, we can check that the error rate increases as the correlation ratio increases. This is something expected, since as more features are correlated, we are likely to lose more information by partitioning features into groups and treating them separately.

For a fixed correlation ratio, say $r = 0.5$ (the midpoints of the four plots in Figure 1), the error rate tends to increase with the number of nodes. This will be also due to the loss of information by partitioning. But it seems that such loss could be compensated by providing nodes some "overlapping" features from other nodes,

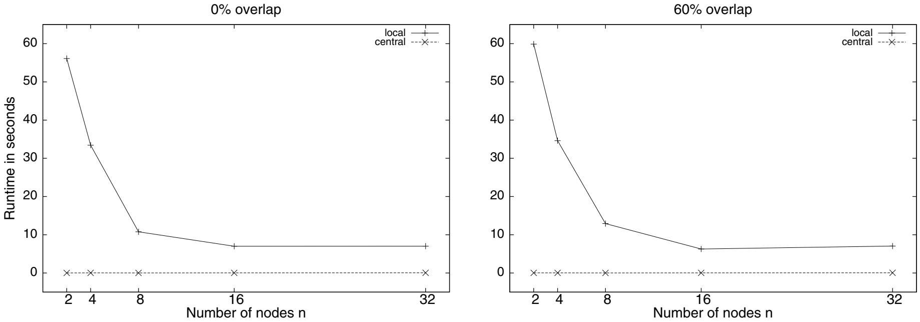


Fig. 2. Runtime (in seconds) at the central node and average runtime (in seconds) for the local optimizations, for different overlaps of features (100%: all features are available at each local node) and numbers of nodes. The correlation ratio is fixed at $r = 0.25$. All values are averages over 20 runs with random splits of features and random approximations of φ.

which can be observed as we scan through the top left, top right, bottom left, and bottom right plots, for 32 nodes with correlation ratio $r = 0.5$ for instance.

Scalability Using Multiple Nodes. Figure 2 shows the average runtime (in seconds) taken in the local and the central optimization of our algorithm. Since the runtime values are not affected much by the correlation ratio r, we show only the plots for $r = 0.25$ here.

The runtime for the local optimization keeps improving as we use more nodes up to $n = 16$, since all optimizations can be done in parallel (the machine had 32 physical cores). Considering the test error rates reported in the corresponding plots (top left and bottom right) in Figure 1, at correlation ratio $r = 0.25$, using more nodes would not harm too much the prediction performance. An exception will be $n = 32$, which seems to make each group too small, deteriorating both scalability and accuracy by a noticeable amount. The runtime for central optimization was almost negligible in this case. The optimization took slightly longer for 60% overlap than the case of no overlap, since the former had to handle larger number of attributes.

5.3 Benchmark on Real-World Data Sets

For benchmark we fix the number of nodes to $n = 8$, since it has showed a good accuracy and speed tradeoff in the experiments with our artificial data. In local optimization, we set the number of SGD iterations to $10m$, ten times of the number of training examples.

Table 3 shows the runtime and test error rate values over 20 repeated runs, except for `CCAT` where we use 12 runs due to its long runtime, of all methods for the five benchmark data sets, without and with 25% overlap of features. In each run we randomize the partitioning of features and the projections for constructing approximate feature mappings, if applicable. For `Single` ASSET, the overlap parameter

Table 3. Training CPU time (in seconds, h:hours) and test error rate (mean and standard deviation %) in parentheses. The features of input vectors are distributed in two different ways: (i) each node contains disjoint set of features (no overlap), or (ii) each node has 25% of its features as copies from other nodes (25% overlap).

No Overlap	ASSET-SVM				SVMLight
	Separated		Composite	Single	
	+ Central	− Central			
ADULT	28 (20.0±0.02)	9 (20.6±1.17)	44 (15.5±0.64)	235 (15.7±0.74)	966 (15.1)
MNIST	101 (11.1±0.39)	87 (11.1±0.39)	539 (7.0±0.72)	1539 (7.0±0.45)	1031 (1.1)
CCAT	1h (26.3±1.00)	1h (26.3±1.00)	8h (20.9±0.63)	-	2h (4.2)
IJCNN	67 (9.1±0.88)	20 (9.5±0.07)	86 (4.1±0.53)	177 (1.6±0.13)	687 (0.7)
COVTYPE	234 (29.3±2.76)	82 (35.2±1.05)	373 (21.1±0.61)	938 (18.0±0.78)	23h (7.4)

25% Overlap	ASSET-SVM				SVMLight
	Separated		Composite	Single	
	+ Central	− Central			
ADULT	28 (19.0±1.69)	9 (19.7±1.34)	48 (15.5±0.49)	235 (15.7±0.74)	966 (15.1)
MNIST	112 (11.1±0.39)	94 (12.1±0.72)	486 (7.3±0.50)	1539 (7.1±0.45)	1031 (1.1)
CCAT	2h (29.5±1.01)	2h (29.5±1.01)	10h (23.8±0.80)	-	2h (4.2)
IJCNN	75 (8.6±1.05)	20 (9.4±0.63)	107 (3.8±0.56)	177 (1.6±0.13)	687 (0.7)
COVTYPE	219 (29.6±2.76)	94 (33.7±1.34)	466 (20.8±0.63)	938 (18.0±0.78)	23h (7.4)

has no effect, so the results are copied in both tables for readability. The results for `Single` on `CCAT` are unavailable for its impractically long runtime.

Gap from the Separation of Optimization. We first compare the results of `Separated` and `Composite` in Table 3. The difference here occurs because of our construction of separable surrogate objective functions using the convex inequality in (7).

Comparing to the third column (`Composite`) of Table 3, the test error rates in the second column (`Separated` without central optimization) has been increased by 1.63 (no overlap) and 1.65 (25% overlap) times on average. Considering that `Composite` has the access to all feature information, through a single composite kernel consisting of all localized kernels, such increments seem to be moderate. In `Separated`, features and optimizations are distributed among the nodes, and thereby the SVM can be solved in much shorter time (about 5 times faster on average) but sacrificing accuracy.

Improvement by Central Optimization. The first two columns of Table 3 show the potential improvement and cost of an additional central optimization. The improvements in test error rates seem to be marginal ($6 \sim 7\%$), but recall that these are obtained using a small subsample (5000) from each training set for the central problem, rather than using the entire set, simulating a limited communication bound.

Gap from Using Localized Kernels. In the third and fourth columns of Table 3, the information of all features is accessed through either a composite kernel consisting of local kernels (Composite), or a single kernel using all features directly (Single). The Composite approach is very similar to the standard MKL, except that here we are using fixed weights among the local kernels. As we can see, the performance in terms of error rates is not very different in these two approaches.

The overall runtime of Composite is shorter than Single, although they have essentially the same time complexity. The savings in Composite might have come from the fact that the input vectors are sparse in our benchmark sets, where subvectors of them tend to be more sparse, reducing the time for projections on random directions in constructing approximate feature mappings.

Gap from Approximating Kernels. The difference in the last two columns of Table 3 is resulted from that the feature mappings of kernels are approximated in Single, whereas SVMLight uses exact kernels. The error rates of Single are comparable in most cases, except for CCAT and COVTYPE. We believe the results will improve with larger approximation dimensions in general. Also, we can consider using different types of approximations for CCAT, and using more iterations in the local optimization for COVTYPE. We refer to [16,11] for more extensive comparison in this respect.

6 Conclusion

We suggest a separable optimization framework for solving the support vector machines on distributed sensor measurements, minimizing communication cost to construct a global predictor. While sacrificing some accuracy, our framework provides a transparent way to derive a separable function that becomes an upper bound of the original SVM objective, based on the convexity of loss functions and approximations to kernel feature mappings.

Acknowledgements. The authors acknowledge the support of Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, project B3 and C1.

References

1. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the 21st International Conference on Machine Learning (2004)
2. Bi, J., Zhang, T., Bennett, K.P.: Column-generation boosting methods for mixture of kernels. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 521–526 (2004)
3. Drineas, P., Mahoney, M.W.: On the nystrom method for approximating a gram matrix for improved kernel-based learning. Journal of Machine Learning Research 6, 2153–2175 (2005)

4. Fine, S., Scheinberg, K.: Efficient svm training using low-rank kernel representations. Journal of Machine Learning Research 2, 243–264 (2001)
5. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer (2001)
6. Joachims, T., Finley, T., Yu, C.-N.: Cutting-plane training of structural svms. Machine Learning 77(1), 27–59 (2009)
7. Joachims, T.: Making large-scale support vector machine learning practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods - Support Vector Learning, ch. 11, pp. 169–184. MIT Press, Cambridge (1999)
8. Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A.: ℓ_p-norm multiple kernel learning. Journal of Machine Learning Research 12, 953–997 (2011)
9. Lanckriet, G.R.G., De Bie, T., Cristianini, N., Jordan, M.I., Noble, W.S.: A statistical framework for genomic data fusion. Bioinformatics 20(16), 2626–2635 (2004)
10. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. In: Proceedings of the 19th International Conference on Machine Learning (2002)
11. Lee, S., Wright, S.J.: ASSET: Approximate stochastic subgradient estimation training for support vector machines. In: International Conference on Pattern Recognition Applications and Methods (2012)
12. Lippi, M., Bertini, M., Frasconi, P.: Collective Traffic Forecasting. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part II. LNCS, vol. 6322, pp. 259–273. Springer, Heidelberg (2010)
13. Morik, K., Bhaduri, K., Kargupta, H.: Introduction to data mining for sustainability. Data Mining and Knowledge Discovery 24(2), 311–324 (2012)
14. Nemirovski, A., Juditsky, A., Lan, G., Shapiro, A.: Robust stochastic approximation approach to stochastic programming. SIAM Journal on Optimization 19(4), 1574–1609 (2009)
15. Nemirovski, A., Yudin, D.B.: Problem complexity and method efficiency in optimization. John Wiley (1983)
16. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Advances in Neural Information Processing Systems, vol. 20, pp. 1177–1184. MIT Press (2008)
17. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: More efficiency in multiple kernel learning. In: Proceedings of the 24th International Conference on Machine Learning (2007)
18. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. Mathematical Programming, Series B 127(1), 3–30 (2011)
19. Sonnenburg, S., Rätsch, G., Schäfer, C., Schölkopf, B.: Large scale multiple kernel learning. Journal of Machine Learning Research 7, 1531–1565 (2006)
20. Wahba, G.: Splines Models for Observational Data. CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 59. SIAM (1990)
21. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th International Conference on Machine Learning, pp. 928–936 (2003)

Unsupervised Inference of Auditory Attention from Biosensors

Melih Kandemir[1], Arto Klami[1], Akos Vetek[2], and Samuel Kaski[1,3]

[1] Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University
[2] Nokia Research Center
Media Technologies Lab
[3] Helsinki Institute for Information Technology HIIT
Department of Computer Science
University of Helsinki

Abstract. We study ways of automatically inferring the level of attention a user is paying to auditory content, with applications for example in automatic podcast highlighting and auto-pause, as well as in a selection mechanism in auditory interfaces. In particular, we demonstrate how the level of attention can be inferred in an unsupervised fashion, without requiring any labeled training data. The approach is based on measuring the (generalized) correlation or synchrony between the auditory content and physiological signals reflecting the state of the user. We hypothesize that the synchrony is higher when the user is paying attention to the content, and show empirically that the level of attention can indeed be inferred based on the correlation. In particular, we demonstrate that the novel method of time-varying Bayesian canonical correlation analysis gives unsupervised prediction accuracy comparable to having trained a supervised Gaussian process regression with labeled training data recorded from other users.

Keywords: Affective computing, Auditory attention, Canonical correlation analysis.

1 Introduction

Attention to external stimulation is a central element in human cognition. By selectively focusing on specific aspects of the stimulation we can control the information gain, to maximally utilize the limited information channels. In Human-Computer Interaction (HCI), attention plays several roles: Information in the user interface should be structured to capture users attention by making it salient when it needs attention [22], but it is also possible to use the attention of the user as a form of implicit input. For visual attention, eye-tracking devices provide a direct interface for measuring attention; they have been used in a range of attentive interfaces, starting from eye tracking based zooming of

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 403–418, 2012.

windows (for a review of attentive interfaces see [25]) to using eye tracking for estimating aspects such as topical relevance in information retrieval [15,20].

Here we venture beyond visual attention to auditory attention. For vision the eye-tracking devices provide relatively direct access to the target of the attention, which has enabled the extensive works on utilizing the attention target as part of the interface design. For auditory attention, however, detecting even where the user is paying attention is largely an open issue, and no simple hardware solutions exist for recording it. In particular, the best current methods are based on direct recording of neuronal activity using functional MRI [16,19] and MEG [12], which are by no means feasible for human-computer interaction, or full-scalp EEG (see [11] for an early example) which is also impractical. For a good overview of auditory attention and extensive list of references, see [7].

In this work we will discuss machine learning approaches useful for creating more portable auditory attention detection devices. Due to the general difficulty of the task, we will consider the simplified task of estimating *how much* a person is attending to particular auditory content. The approach could be directly generalized to the task of estimating to which of multiple parallel auditory streams the user is focusing on, by comparing the level of attention paid to each of the streams, but to simplify the experimental setup we consider explicitly only the task of measuring the amount of attention for a single source.

While specific hardware focusing on auditory attention is lacking, we revert to the choice of using a combination of available physiological sensors for recording the state of the user. We record neuronal activity with an easy-to-wear single-channel EEG, the amount of body movement with an accelerometer, and eye movements with an eye-tracker. While these sensors are clearly not optimal for detecting auditory attention, they still provide multivariate signals that represent the activity of the individual user while she is listening to some auditory content. The field of affective computing studies the use of such signals for inferring various cognitive and affective properties of the user, and relatively good success has been demonstrated for instance in inferring emotional valence and arousal [5,18], specific emotions [13], and mental workload [28]. Hence, it is a reasonable assumption that we could get a handle on the attention with similar sensors as well. In fact, [17] has already demonstrated success in discovering loss of auditory attention due to external interruptions by monitoring the galvanic skin response.

Given the sensory signals, the task of detecting auditory attention is, in principle, a straightforward learning task. We merely need to obtain ground truth training labels and train a classifier or a regression model for inferring the labels from the signals. For a classifier, the labels would be high vs. low attention, and for a regression model the actual level of attention. However, it is extremely challenging to collect training labels for the task of inferring the level of attention. For example, if the subject is listening to a music piece, we cannot ask him to continuously rate his level of attention since providing that feedback would change his behavior; needing to provide the evaluation would prevent him from naturally attending to the music. It is also unreasonable to expect that people would be able to quantify their level of attention to arbitrary auditory contents after the experiment with high accuracy.

The only remaining way of collecting the training labels is to conduct a laboratory experiment where the labels stem from a controlled experiment. In this work, we present a simple experiment of that sort, using an additional visual task of varying complexity to control the level of attention remaining for an auditory listening task. While such a procedure gives training labels, it is important to realize that it will only provide them for the users who took part in the particular laboratory experiment; it still remains infeasible to obtain any training data whatsoever for the eventual users of an auditory attention detector. This observation implies that any model inferring the level of attention from the sensory signals must be user-independent. Such models, in turn, are known to be of relatively poor accuracy due to considerable user-specific variation in the sensory signals. Nevertheless, in this work we demonstrate that we can infer the level of auditory attention with reasonable accuracy using user-independent supervised models, by applying two state-of-art probabilistic kernel-based regressors: Gaussian process regression [21] and Relevance Vector Machines [23].

Our main contribution, however, is an alternative way of inferring the level of attention that does not require any training labels whatsoever. Instead, we make a hypothesis that the amount of synchrony or correlation between the physiological signals and the auditory content is modulated by the level of attention. That is, we assume that any signals recorded from a user not paying attention to the audio will be independent of the audio signal, whereas high degree of attention is reflected as increased correlation between some of the physiological signals and the audio content. Assuming the hypothesis holds, we can directly detect auditory attention as correlation between the two signals, without needing any training data. To further illustrate the approach and its relationship with the supervised one, the analysis pipelines for both are depicted in Figure 1.

We measure the correlation with canonical correlation analysis (CCA) and its Bayesian re-formulation as a latent variable model [2,27]. Using the Bayesian formulation not only helps with limited amount of data, but enables encoding prior knowledge on the underlying signals into the model. In this work we utilize the fact that the measurements are time series, and introduce a novel time-dependent Bayesian CCA model by encoding time-dependent interactions in the generative description. We learn the model from the coupled physiological signals and features computed for the audio content, and then measure the amount of correlation to represent the level of attention. We demonstrate that the correlation reveals the level of attention with accuracy comparable to the user-independent supervised models. The empirical experiments hence demonstrate that we can infer the level of attention from physiological signals, and more importantly that we can do it without requiring any labeled training data at all.

We start the rest of the paper by first introducing some prototypical application scenarios for auditory attention detection, providing a context and motivation for the more technical sections. We then proceed to explain the computational models needed both for supervised user-independent inference of attention and for

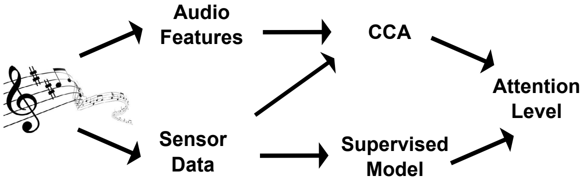


Fig. 1. Illustration of the two modeling pipelines for estimating the level of attention from the biosensors. The unsupervised approach evaluates the correlation between the sensor data and the audio content, and uses the inverse correlation as a predictor for the level of attention. The supervised approach uses only the sensor data, and applies a model learned from other users for predicting the level of attention.

measuring the amount of correlation between the physiological signals and the auditory content. After describing the models we explain the empirical experiment conducted for recording data to train and evaluate the models, and then show the empirical results demonstrating the accuracy of the proposed methods.

2 Application Overview

Albeit we here consider the task of inferring the level of auditory attention primarily as a basic research question, it has several direct application possibilities that are worth highlighting. A simple example would be an auto-pause tool for audio players; the attention recognizer would be running continuously on the background and whenever it recognizes that the level of attention is particularly low it pauses the audio automatically so that the user can continue listening for the audio after the interfering concurrent task is over. Alternatively, the tool can simply keep track of the moments with low attention, allowing the user to easily return to them later (for a practical example, see [17]), for example to re-cap details of a technical description in a podcast they might have missed during the first listening.

There are also applications where storing the moments with the *highest* attention could be useful. Such an automatic highlighting tool could capture, for example, the moments when the user most enjoyed a piece of music. Those pieces could even serve as a query to a music retrieval engine; the user could carry out a search for other songs similar to the most enjoyable parts of the song he just listened. Besides static content, such as music or podcasts, the tool could also be used for highlighting more dynamic content. For example, it could be used to summarize a meeting as a combination of the moments where reasonable amount of attention was paid to the discussion.

In another application scenario the goal is to detect the attention target. In an environment with multiple overlapping auditory streams, we can measure the amount of correlation with respect to each of the sources and detect the target of primary attention as the one with the highest correlation. This enables building for instance auditory interfaces where attention is used to implicitly select one out of multiple alternatives.

3 Modeling Dependencies

We first describe the classical method for estimating the amount of multivariate correlation between two data sources, the canonical correlation analysis (CCA). We then proceed to describe the Bayesian variant of CCA, following the formulation in [27], which makes CCA applicable for high-dimensional data and allows various extensions. Finally, we introduce the novel time-dependent Bayesian CCA model that explicitly models continuity in time-series data.

3.1 CCA

Given two data matrices, $\mathbf{X} \in \mathbb{R}^{N \times D_x}$ and $\mathbf{Y} \in \mathbb{R}^{N \times D_y}$, with N samples (rows) and D_x and D_y features (columns), the CCA finds linear projection weights $\mathbf{u} \in \mathbb{R}^{1 \times D_x}$ and $\mathbf{v} \in \mathbb{R}^{1 \times D_y}$ such that the Pearson correlation

$$\rho = \mathrm{cor}(\mathbf{X}\mathbf{u}^T, \mathbf{Y}\mathbf{v}^T)$$

is maximized. Since correlation is invariant to the scale, the norms of $\mathbf{u}$ and $\mathbf{v}$ can be fixed to unity. The above formulation defines the most correlating one-dimensional subspace; further components indexed by a subscript can be obtained by adding an orthogonality constraint $cor(\mathbf{X}\mathbf{u}_k^T, \mathbf{X}\mathbf{u}_l^T) = 0$ for all k and l (and similarly for $\mathbf{Y}$). In practice, we can readily compute $\min(D_x, D_y)$ canonical correlations ρ_k and the associated projections $(\mathbf{u}_k, \mathbf{v}_k)$ by solving a single generalized eigenvalue problem (see for, instance, [10] for details).

While CCA is typically used for gaining an understanding of the correlations between the two data sets (by interpreting $\mathbf{u}$ and $\mathbf{v}$), it readily provides a measure for the *amount of dependency* between them. We summarize the dependency with the quantity

$$I(\mathbf{X}, \mathbf{Y}) = -\frac{1}{2} \sum_{k=1}^{\min(D_x, D_y)} \log\left(1 - \rho_k^2\right),$$

which corresponds to the mutual information between the two sources if they are jointly multivariate normal. For non-normal data the quantity does not correspond to the mutual information, but is still a good estimate of the total dependency, summarizing all correlations into a single number.

In our application, the task is to measure the amount of correlation for a subset of the samples. We do this by first learning the model to maximize the correlation over the whole available data. Given the model (the projections), we then evaluate the correlation for any subset L of the samples by simply estimating the correlations (and the above mutual information summary) between $\mathbf{X}_L\mathbf{u}_k^T$ and $\mathbf{Y}_L\mathbf{v}_k^T$.

3.2 Bayesian CCA (BCCA)

While CCA is a straightforward method with guaranteed convergence to a global optimum, it has a number of shortcomings that can be addressed by switching to the probabilistic framework. First of all, CCA is prone to overfitting to high-dimensional data, and especially for $N < \min(D_x, D_y)$ necessarily returns correlations of exactly one due to linear dependency of the features. For preventing this we need proper regularization, which we implement through priors and variational inference in the Bayesian framework. This particular choice gives us also another advantage: It allows extending the model by making slight changes to the generative model of CCA (see for instance [1,8,26] for examples). After recapping the model, we will in the next section show how the Bayesian treatment of CCA enables incorporating time-dependencies between the samples, a property that would not be easy to add in the original linear algebraic formulation.

The Bayesian CCA builds upon the probabilistic interpretation of CCA by [2]. The basic idea is that the two data sources are generated from a common latent representation with a linear transformation, with arbitrary additive Gaussian noise independent of the other source. More formally,

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mathbf{x} &\sim \mathcal{N}(\mathbf{W}_x \mathbf{z}, \boldsymbol{\Psi}_x) \\ \mathbf{y} &\sim \mathcal{N}(\mathbf{W}_y \mathbf{z}, \boldsymbol{\Psi}_y), \end{aligned} \tag{1}$$

where $\mathbf{z} \in \mathbb{R}^{1 \times K}$ is a K-dimensional latent signal and $\mathbf{W}_x \in \mathbb{R}^{D_x \times K}$ and $\mathbf{W}_y \in \mathbb{R}^{D_y \times K}$ are projections mapping the latent signals to the observations. The noise covariances $\boldsymbol{\Psi}_x$ and $\boldsymbol{\Psi}_y$ model the variation independent of the other sources. In practice, especially for high-dimensional data, we need to assume low-rank noise covariances $\boldsymbol{\Psi}_x$ and $\boldsymbol{\Psi}_y$ to prevent needing to make inference over the $D_x \times D_x$ and $D_y \times D_y$ covariance matrices, which leads to the Bayesian CCA formulation of [27]:

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ [\mathbf{x}; \mathbf{y}] &\sim \mathcal{N}(\mathbf{W}\mathbf{z}, \boldsymbol{\Sigma}), \end{aligned}$$

where $\boldsymbol{\Sigma}$ is a block-diagonal matrix $\boldsymbol{\Sigma} = [\sigma_x^2 \mathbf{I}, \mathbf{0}; \mathbf{0}, \sigma_y^2 \mathbf{I}]$. By making $\mathbf{W}$ group-wise sparse with the sparsity-inducing prior

$$\begin{aligned} \beta_{xk} &\sim \mathcal{G}(\alpha_0, \beta_0) \\ \beta_{yk} &\sim \mathcal{G}(\alpha_0, \beta_0) \\ p(\mathbf{W}) &= \prod_{k=1}^{K} \left(\mathcal{N}(\mathbf{W}_x(k) | \mathbf{0}, \beta_{xk}^{-1} I) \mathcal{N}(\mathbf{W}_y(k) | \mathbf{0}, \beta_{yk}^{-1} I) \right), \end{aligned}$$

where $\mathcal{G}(\alpha_0, \beta_0)$ is a flat Gamma distribution ($\alpha_0 = \beta_0 = 10^{-14}$), we will get projections $\mathbf{W}$ that factorize as

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_x & \mathbf{V}_x & \mathbf{0} \\ \mathbf{W}_y & \mathbf{0} & \mathbf{V}_y \end{bmatrix}.$$

After marginalizing out the latent components corresponding to the columns of $\mathbf{W}$ having a zero block for either data source, induced by the group-wise sparsity prior, the above model becomes equivalent to (1) with $\boldsymbol{\Psi}_X = \mathbf{V}_x\mathbf{V}_X^T + \sigma_x^2\mathbf{I}$ and $\boldsymbol{\Psi}_y = \mathbf{V}_y\mathbf{V}_y^T + \sigma_y^2\mathbf{I}$. In summary, the resulting model implements the Bayesian CCA model with a low-rank assumption for the noise covariances within each data source, but does not require specifying the rank of either the correlating subspace or the noise covariances in advance. Instead, they will all be learned automatically from the data.

We do the inference using a variational approximation, assuming the posterior $p(\Theta|\mathbf{X},\mathbf{Y}) = p(\sigma_x^2, \sigma_y^2, \beta_x, \beta_y, \mathbf{Z}, \mathbf{W}|\mathbf{X},\mathbf{Y})$ can be approximated by a factorized distribution

$$Q(\Theta) = q(\sigma_x^2)q(\sigma_y^2)\prod_{k=1}^{K} q(\beta_{xk})q(\beta_{yk})\prod_{i=1}^{N} q(\mathbf{z}_i)\prod_{d=1}^{D} q(\mathbf{W}_{d.}),$$

and minimizing the Kullback-Leibler divergence between $Q(\Theta)$ and $p(\Theta|\mathbf{X},\mathbf{Y})$. This results in a set of mean-field equations updating each term $q(\cdot)$ at a time until convergence to a local optimum; the details can be found in [27].

For evaluating the correlation we estimate the conditional densities $p(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z}|\mathbf{y})$ and then compute the correlation between their expectations. This can be done for any subset of the samples using a procedure similar to updating $q(\mathbf{Z})$ while learning the posterior approximation.

3.3 Time-Dependent Bayesian CCA (T-BCCA)

One advantage of the Bayesian formulation for CCA is that it allows easily extending the model to take into account particular properties of the underlying data. As practical examples, in [1,26] more robust variants were introduced by replacing the normal distributions with t-distributions, in [14,26] mixtures of CCAs, and in [8] sparse variants. In this paper, we will extend Bayesian CCA to a state-space model that is more accurate for modeling correlations between two multivariate time series.

The key idea of the novel time-dependent CCA is that the latent variables z will have a Markovian assumption. Instead of drawing each z_t independently from the same prior, we introduce the prior

$$\begin{aligned}\mathbf{z}_0 &\sim N(\mathbf{0},\mathbf{I})\\ \mathbf{z}_t &\sim \mathbf{T}\mathbf{z}_{t-1} + N(\mathbf{0},\sigma_0^2\mathbf{I})\end{aligned}$$

where $\mathbf{T}$ governs the evolution of the latent space and σ_0^2 controls the amount of stochastic noise.

We retain the variational Bayesian framework for inference, and are able to reuse the update formulas for the various terms in the approximation except for $q(\mathbf{Z})$. For that, we learned a Kalman filter along with the Rauch-Tung-Striebel smoother [9], using a forward-backward procedure as described in [3]. Since the Bayesian CCA assumes independent latent components, we restrict $\mathbf{T}$ to be diagonal to avoid modeling dependencies between them and use variational inference with prior centered around the identity matrix. Furthemore, we set $\sigma_0^2 = 1$ to fix the scale, but could do variational inference over this parameter as well.

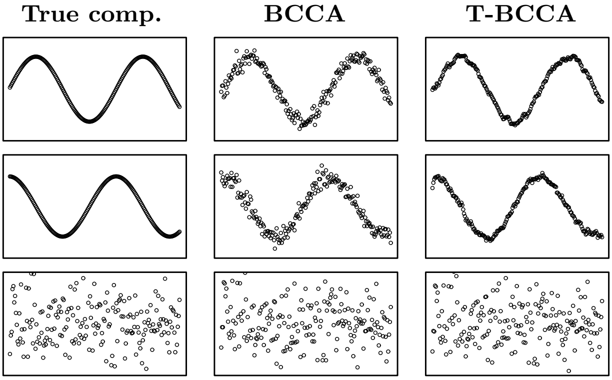


Fig. 2. Illustration of the importance of modeling time-dependencies in CCA modeling. The left column shows three latent components underlying a generated data set, the middle column shows the latent components estimated by regular Bayesian CCA, and the right column shows the estimated components when modeling also the time-dependency. We see that already the regular Bayesian CCA (BCCA) captures the signals roughly correctly, but that the time-dependent model (T-BCCA) gives much more accurate estimates for the sinusoidal signals. The component with no time-dependencies (bottom row) is modeled equally well; T-BCCA learns to automatically set the corresponding element in **T** to zero.

To briefly illustrate the advantage of modeling time-dependencies in the latent space, we applied both the regular Bayesian CCA model described in the previous section (which corresponds to $\mathbf{T} = 0$ and $\sigma_0^2 = 1$ in the more general formulation) and the time-dependent model to a simple simulated data. The data has three latent components of which some show clear time-continuity, and as shown in Figure 2 modeling the time-series nature results in more accurate estimates for them.

4 Supervised Learning

Given the available sensor data, the usual approach for inferring the level of attention would be to use supervised learning. That is, we would collect labels for training instances and train a classifier or regressor for predicting the attention. As discussed in the Introduction, gathering such labeling data for the task of auditory attention is extremely challenging and the only reasonable way is to use laboratory experiments with controlled stimulation. This allows gathering training data from laboratory users, but not from the eventual users of an auditory attention predictor system. Hence we require user-independent models in this task.

In this paper, we will compare the unsupervised approach with supervised learning where the model is learned from a training corpus measured and labeled for other users. We chose two state-of-art supervised methods: Relevance Vector Machine (RVM) [23] and Gaussian process (GP) regression [21], as representative alternatives. Both are probabilistic kernel-based learning methods that enable non-linear mappings from the input to the level of attention. To capture non-linearity, we used the Radial Basis Function (RBF) kernel:

$$k(\mathbf{x}, \mathbf{x'}) = \exp(\frac{\|\mathbf{x} - \mathbf{x'}\|^2}{2\nu^2})$$

with both of these methods. For each method, we learned the kernel parameter ν using type II maximum likelihood.

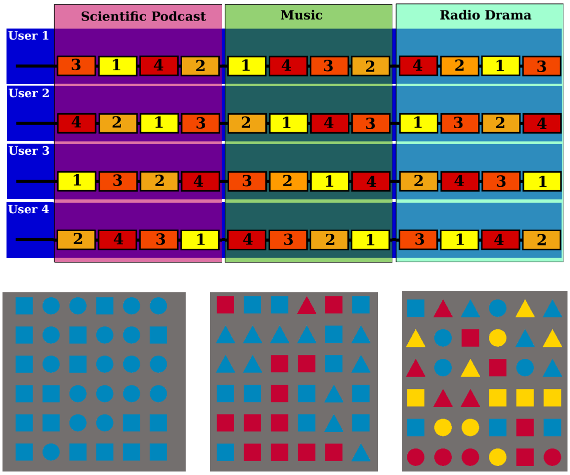
5 Experimental Setup and Data

To train and evaluate the proposed methodology for inferring the level of auditory attention, we created a simple laboratory experiment. The subjects listen to three types of audio content (scientific podcast, popular music, and audio drama) while being measured with three different sensors (NeuroSky single-channel EEG device, accelerometer measuring body movement, and eye-tracker measuring the pupil dilation). Their level of attention to the auditory content is controlled by a simultaneous alternative task with tunable difficulty competing for their attentional resources. Based on the limited-capacity theorem asserting that there is a direct performance tradeoff between simultaneous auditory and visual tasks [4], we assume that the auditory attention is low whenever the user is paying a high level of attention to the alternative task, and vice versa.

For the alternative task we chose a visual search task called *conjunction search* where the user searches for objects identified by multiple features [24]. The user is presented a grid of items, and asked to tell whether any of the items on the screen is unique in terms of color and shape. We assigned the user the binary detection task, instead of asking him to point where the unique item is, to avoid introducing unnecessary movements.

The visual task was presented in four difficulty levels. We tuned the difficulty of the search task by the number of different colors and shapes of the objects. The easiest level (level 1) was simply the blank screen, hence there was no search task at all. The remaining three levels of difficulty had 2, 4, and 9 different kinds of objects, respectively (see the bottom half of Figure 3 for illustration of the stimuli). This provides data with four ground-truth levels of visual attention, and we assume that the auditory attention has an inverse monotonous relationship to visual attention.

We constructed a partially balanced experimental setup for our 12 voluntary test users (7 male and 5 female university students aged from 22 to 29 years). All of them listened to the three audio contents, a scientific podcast, music, and radio drama, of 4 minutes each. There was 1 minute of each visual task level within each audio type. The order of the visual task levels within each audio type was balanced across users using the 4 × 4 Latin squares design. The order



	Scientific Podcast				Music				Radio Drama			
User 1	3	1	4	2	1	4	3	2	4	2	1	3
User 2	4	2	1	3	2	1	4	3	1	3	2	4
User 3	1	3	2	4	3	2	1	4	2	4	3	1
User 4	2	4	3	1	4	3	2	1	3	1	4	2

Fig. 3. Top: Illustration of the experimental setup. The subjects listened to three types of audio content while performing visual search tasks of varying difficulty (four levels). The blocks of the visual tasks occurred at the same time for all subjects, but the design was balanced so that the difficulty levels were in a different order for different subjects, as well as in a different order between the different audio types for each subject. Furthermore, the subjects listened the three audio types in different orders (not shown in the image for clarity). **Bottom:** Examples of the three visual search task difficulties corresponding to levels 2, 3, and 4 from left to right. Level 1 is blank screen where there is no visual search task at all.

of the audio contents was also balanced according to another 3×3 Latin squares. Figure 3 illustrates the course of the experiment.

For the data analysis we processed both the auditory content and the biosignals into vectorial samples, forming each sample from a 250ms contiguous block of the signals. The music is represented by 17 numerical features capturing primarily timbral and rhytmic properties of the music, computed using the MIR toolbox [6]. The idea is that the representation would capture essential characteristics of the audio content. The physiological signals, in turn, were summarized through several features stemming from the affective computing literature, considered to be reasonable approximations of the information content in the physiological signals. The actual features used are listed in Table 1.

6 Results

For evaluating the accuracy and feasibility of the proposed attention inference method, we ran two separate computational experiments on the experimental data described in the previous section. Here we both describe the experiments and report their results.

Table 1. Top: Physiological features extracted from the data collected by three sensors: accelerometer, eye-tracker, and single-channel EEG. **Bottom:** Audio features computed by the MIR toolbox [6].

Physiological features
3D body motion and pupil diameter: mean and standard deviation mean of the derivative, mean, median, and maximum peak-to-peak interval
Single-channel EEG: spectral power in (0.5–2.75) Hz, (3.5–6.75) Hz, (7.5–9.20) Hz,(10.0–11.75) Hz, (13.0–16.75) Hz,(18.0–29.75) Hz, (31.0–39.75) Hz, (41.0–49.75) Hz

Audio features
zero-crossing rate, spectral centroid, brightness, spectral spread, kurtosis, MFCC (Mel-frequency cepstral coefficients), skewness, roll-off, entropy, spectral-flatness, roughness, RMS (root-mean-square), spectral flux, novelty of spectral flux, fluctuation, fluctuation centroid, fluctuation entropy

6.1 Experiment 1: Inferring the Level of Attention for Long Time Blocks

In the first experiment we study the problem of inferring the level of attention for each of the experimental blocks. This answers the question of whether we can differentiate between different levels of attention during periods of time lasting roughly one minute each. The results would be directly applicable to scenarios such as meeting highlighting but would not be sufficient for choosing the attention target in an interactive interface, for instance.

We analyze each of the audio types and users separately, resulting in a total of $12 \times 3 = 36$ models. For the correlation-based models we learn the CCA using all the data for that user-audio pair and then evaluate the correlation for each of the blocks corresponding to one level of ground truth attention. Since we have four levels of ground truth attention, this gives us four correlation scores which we sort in the decreasing order to predict the attention. For each user-audio pair we then compute the accuracy as the number of correct ranks with respect to the ground truth. For example, if the block with the hardest visual task is ranked last, the score increases by one. This measure is equivalent to classification accuracy for a scenario where we know that each class occurs only once in the test set.

For the supervised models we train a regression model with the labeled training data for all other users and then apply it to the four blocks of the user in question. This is done separately for each audio type, and we again rank the resulting regression scores to label the four blocks with the levels of attention. That is, we use the exactly same measure as for the correlation-based models.

Table 2 collects the average scores (over users, normalized so that 1 equals to a perfect result) for all of the methods and all three audio types. The supervised user-independent approaches provide the best results, but the unsupervised variants closely follow with only marginally lower accuracies. Of these three, the

T-BCCA model has an accuracy over 40% for all three audio types, which is a promising signal for real applications. To evaluate the reliability of the results we performed Wilcoxon signed rank test over the results of all three audio types (to obtain more statistical evidence over the $12 \times 3 = 36$ independent scores), revealing that all five methods are significantly ($p < 0.05$) better than the random baseline, but that the differences between the alternatives are not significant.

Note that in this experiment we normalized the correlations by dividing them by the mean of the correlations for all users during the same audio content. This was done to reduce the potential bias caused by the properties of the auditory content itself. It is easy to imagine that certain types of audio content, for example catchy music pieces, could result in naturally higher correlation levels with the sensory signals for all attention levels. The kind of normalization done in this experiment could be done for real applications given access to sensory data of other users having listened to the same content, still without needing any labeled data. As this is not necessarily the case in many situations, we also re-ran the experiment without such normalization. This results in a drop of (on average) two percentage points for each of the unsupervised methods. Together these two experiments indicate that it pays off to remove the content-specific effect on the correlation, but that it is not absolutely crucial and the methods work even without any earlier data from other users.

Table 2. The classification accuracy for detecting the four levels of attention in the experiment. All five methods outperform the chance level with statistical significance ($p < 0.05$; Wilcoxon). The noteworthy observation is that the unsupervised CCA-based methods are only slightly worse than the supervised ones (GP and relevance vector regression).

Method	Scientific Podcast	Music	Radio Drama	Average
Gaussian Process regression	0.52	0.38	0.50	0.47
Relevance Vector regression	0.52	0.44	0.42	0.46
Time-dependent Bayesian CCA	0.42	0.46	0.44	0.44
Bayesian CCA	0.25	0.52	0.44	0.40
Classical CCA	0.35	0.44	0.48	0.42
Random baseline	0.25	0.25	0.25	0.25

6.2 Experiment 2: Inferring Short-Term Attention

The above experiment considered the problem of inferring the level of attention for time-periods lasting roughly one minute, and also matched exactly the experimental setup of our data. For practical application scenarios we might want to infer the level of attention also for shorter time periods, for example to enable auto-pause or audio highlighting, or to more quickly recognize which of several overlapping audio streams the user is attending to.

In this experiment we study how short we can make the time window while still getting an estimate that is better than random chance. We do this by training

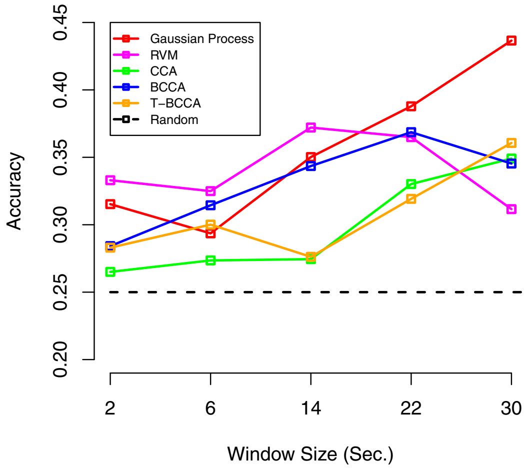


Fig. 4. The classification accuracy of detecting the four levels attention is given for all models in comparison as a function of time window size. For each window size, the scores are averaged over all users and audio types. The unsupervised CCA-based methods are all better than chance, but for most window sizes are slightly beaten by the supervised methods. This figure is best viewed in colors.

the models as above, but making the predictions for all consecutive time windows of certain length[1] instead of the full blocks as we did above. We measure the performance as before, by comparing the true ranking of the M windows with the ranks obtained by ordering the windows based on the correlation score or the regressor output. Again we assume we know how many windows of each attention level we have; this is done for the purpose of measuring only – in practical applications we will always have the full ranking and need not make such assumptions. Figure 4 shows the resulting average accuracy (averaged over both the users and audio types) of the alternative methods as a function of the window size, showing the intuitive trend that inferring the level of attention gets harder when we have less data. Again the supervised predictors are the best, but the unsupervised models also outperform the chance level. For shorter window lengths the Bayesian CCA variants outperform the classical one.

7 Discussion

Visual attention plays a central role in human-computer interaction, and being able to measure the target of the attention with eye-tracking devices has enabled novel types of user interfaces that infer information from the attention [15,20]. Auditory attention, while equally important for the daily life of humans, has been studied much less extensively, not only because auditory interfaces are less

[1] We exclude windows where the ground truth labeling changes during the window.

common but also because no hardware solutions for estimating the level or target of auditory attention exist.

In this work we studied the problem of inferring the level of auditory attention from physiological signals. We compared two alternative approaches for inferring the level: supervised learning with user-independent models, and unsupervised inference based on the assumption that the physiological measurements correlate with the auditory content more strongly when the user is attending to the content. We used state-of-art computational models, Gaussian process [21] and Relevance Vector Machine [23] regression for supervised learning and Bayesian canonical correlation analysis [27] for unsupervised learning, and extended the latter approach to the novel time-dependent BCCA model to better match the underlying time-series nature of the signals. Our experiments demonstrated that both approaches can extract information on the amount of attention paid to three different types of auditory content. The accuracy is not yet sufficient for practical applications, but both approaches outperform chance level with statistical significance, implying that the direction is feasible. The main observation is that the unsupervised methods provide recognition accuracy only slightly worse than that of the supervised models. Even though the time-dependent model was demonstrated to better capture the time-dependencies on artificial generated data, its performance on the real data was only comparable to not modeling the dynamics; the time-dependent model was the best unsupervised variant for long windows, but for short windows the regular Bayesian CCA was better.

One aspect worth noting is that our experiments do not reveal whether the supervised predictors are predicting the level of attention to the auditory content, or merely predicting the attention paid for the visual search tasks. This is because they take as input only the sensory signals and the output labels are equivalent for both tasks. Similar problems are likely to remain for all isolated experiments trying to control the level of auditory attention, and hence for training supervised models guaranteed to address the right aspect one would need to use several alternative techniques for controlling the auditory attention: A supervised model could only be relied to predict the auditory attention itself if it generalizes over all such ways of control. The unsupervised approach, however, does not suffer from the same problem, since we are not learning the parameters to predict the attention but instead are merely estimating the amount of correlation between the sensory signals and the auditory content. This means the approach directly answers to the question of auditory attention, and would not have the flexibility to model alternative explanations for the predicted attention.

For improving the accuracy towards the level required for real-world applications, the most promising direction is to improve the instrumentation and the signal representations. Our main focus was on the machine learning question and the associated computational methods, instead of building a practical attention-detection tool. Replacing the three sensors used in our experiments with sensors more suitable for detecting correlations with the auditory content (for example, a multi-channel EEG additionally recording areas closer to the auditory cortex) should dramatically improve the accuracy, yet the computational methods presented here would remain applicable. Regarding the methods development, a promising direction would be to study methods that allow automatic normalization of the correlation levels with respect to the auditory content. Learning a regressor from the auditory content to the average correlation (independent of the task), would allow normalizing the correlation measures with respect to the content without needing to rely on having measurements from other users.

Acknowledgments. We acknowledge the Academy of Finland (COIN Finnish CoE and decision number 133818) and the UI-ART project for funding. SK and AK were also supported by PASCAL2 European NoE.

References

1. Archambeau, C., Bach, F.: Sparse probabilistic projections. In: Proceedings of NIPS, pp. 73–80 (2009)
2. Bach, F.R., Jordan, M.I.: A probabilistic interpretation of canonical correlation analysis. Tech. Rep. 688, Department of Statistics, University of California, Berkeley (2005)
3. Barber, D., Chiappa, S.: Unified inference for variational Bayesian linear gaussian state-space models. In: Proceedings of NIPS (2006)
4. Bonnel, A.M., Hafter, E.R.: Divided attention between simultaneous auditory and visual signals. Perception & Psychophysics 60(2), 179–190 (1998)
5. Chanel, G., Kronegg, J., Grandjean, D., Pun, T.: Emotion Assessment: Arousal Evaluation Using EEG's and Peripheral Physiological Signals. In: Gunsel, B., Jain, A.K., Tekalp, A.M., Sankur, B. (eds.) MRCS 2006. LNCS, vol. 4105, pp. 530–537. Springer, Heidelberg (2006)
6. Eerola, T., Toiviainen, P.: Mir in matlab: The midi toolbox. In: Proceedings of the International Conference on Music Information Retrieval, ISMIR (2004)
7. Fritz, J., Elhilali, M., David, S., Shamma, S.: Auditory attention–focusing the searchlight on sound. Current Opinions in Neurobiology 17(4), 437–455 (2007)
8. Fujiwara, Y., Miyawaki, Y., Kamitani, Y.: Estimating image bases for visual image reconstruction from human brain activity. In: Procedings of NIPS, pp. 576–584 (2009)
9. Grewal, M.S., Andrews, A.P.: Kalman Filtering: Theory and Practice Using MATLAB. John Wiley and Sons, Inc. (2001)
10. Hardoon, D.R., Szedmak, S., Shawe-Taylor, J.: Canonical correlation analysis: An overview with application to learning methods. Neural Computation 16(12), 2639–2664 (2004)
11. Hillyard, S., Hink, R., Schwent, V., Picton, T.: Electrical signs of selective attention in the human brain. Science 182, 177–180 (1973)
12. Jääskeläinen, I., Ahveninen, P., Bonmassar, G., Dale, A., Ilmoniemi, R., Levanen, S., Lin, F., May, P., Melcher, J., Stufflebeam, S., et al.: Human posterior auditory cortex gates novel sounds to consciousness. Proceedings of National Academy of Science USA 101, 6809–6814 (2004)
13. Kim, J., André, E.: Emotion recognition based on physiological changes in music listening. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(12), 2067–2083 (2008)

14. Klami, A., Kaski, S.: Local dependent components. In: Proceedings of the International Conference on Machine Learning, pp. 425–432. Omnipress (2007)
15. Kozma, L., Klami, A., Kaski, S.: GaZIR: Gaze-based zooming interface for image retrieval. In: Proceedings of the Conference on Multimodal Interfaces (ICMI), pp. 305–312. ACM, New York (2009)
16. Nakai, T., Kato, C., Matsuo, K.: An fMRI study to investigate auditory attention: a model of the cocktail party phenomenon. Magn. Reson. Med Sci. 4(2), 75–82 (2005)
17. Pan, M.K., Chang, G.J.S., Himmetoglu, G.H., Moon, A., Hazelton, T.W., MacLean, K.E., Croft, E.A.: Galvanic skin response-derived bookmarking of an audio stream. In: Proceedings of the Human Factors in Computing Systems (CHI), pp. 1135–1140. ACM, New York (2011)
18. Picard, R.W., Vyzas, E., Healey, J.: Toward machine emotional intelligence: Analysis of affective physiological state. IEEE Trans. Pattern Anal. Mach. Intell. 23(10), 1175–1191 (2001)
19. Pugh, K., Shaywitz, B., Shaywitz, S., Fulbright, R., Byrd, D., Skudlarski, P., Shankweiler, D., Katz, L., Constable, R., Fletcher, J., Lacadie, C., Marchione, K., Gore, J.: Auditory selective attention: An fMRI investigation. Neuroimage 4, 159–173 (1996)
20. Puolamäki, K., Salojärvi, J., Savia, E., Simola, J., Kaski, S.: Combining eye movements and collaborative filtering for proactive information retrieval. In: Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR), pp. 146–153. ACM, New York (2005)
21. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
22. Sharp, H., Rogers, Y., Preece, J.: Interaction Design: Beyond Human-Computer Interaction, 2nd edn. John Wiley and Sons (2007)
23. Tipping, M.E.: The relevance vector machine. In: Proceedings of NIPS. MIT Press, Cambridge (2000)
24. Treisman, A.M., Gelade, G.: A feature-integration theory of attention. Cognitive Psychology 12(1), 97–136 (1980)
25. Vertegaal, R., Shell, J.S.: Attentive user interfaces: the surveillance and sousveillance of gaze-aware objects. Social Science Information 47(3), 275–298 (2008)
26. Viinikanoja, J., Klami, A., Kaski, S.: Variational Bayesian Mixture of Robust CCA Models. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part III. LNCS, vol. 6323, pp. 370–385. Springer, Heidelberg (2010)
27. Virtanen, S., Klami, A., Kaski, S.: Bayesian CCA via group sparsity. In: Proceedings of the International Conference on Machine Learning (ICML 2011), pp. 457–464. ACM, New York (2011)
28. Wilson, G.F., Russell, C.A.: Real-time assessment of mental workload using psychophysiological measures and artificial neural networks. Human Factors 45(4), 635–643 (2003)

A Family of Feed-Forward Models for Protein Sequence Classification

Sam Blasiak, Huzefa Rangwala, and Kathryn B. Laskey

George Mason University

Abstract. Advances in sequencing have greatly outpaced experimental methods for determining a protein's structure and function. As a result, biologists increasingly rely on computational techniques to infer these properties of proteins from sequence information alone. We present a sequence classification framework that differs from the common SVM/kernel-based approach. We introduce a type of artificial neural network which we term the Subsequence Network (SN) that incorporates structural models over sequences in its lowest layer. These structural models, which we call Sequence Scoring Models (SSM), are similar to Hidden Markov Models and act as a mechanism to extract relevant features from sequences. In contrast to SVM/kernel methods, which only allow learning of linear discrimination weights, our feed-forward structure allows linear weights to be learned in conjunction with sequence-level features using standard optimization techniques.

1 Introduction

Advances in sequencing have greatly outpaced experimental methods for determining a protein's structure as well as its role within the complex network of interactions taking place inside living organisms. As a result, biologists increasingly rely on computational techniques to infer structural and functional properties of proteins from sequence information alone.

Popular and successful approaches for protein classification employ Support Vector Machines (SVM) [9,10,11,17,15,2]. Performance of SVM-based classifiers is highly dependent on the kernel function, which can be difficult to specify and to interpret. Kernel functions often have free parameters that must be set either through cross validation or heuristics. Further, ad hoc techniques are often employed to normalize pre-computed kernels so that the algorithm can learn larger margins between classes.

We present a sequence classification framework that differs from the SVM/kernel-based approach. We construct a type of neural network called a Subsequence Network (SN) that incorporates structural models over subsequences. These structural models, called Sequence Scoring Models (SSMs), are similar to Hidden Markov Models and act as a mechanism to extract relevant features from sequences. Our feed-forward structure allows standard optimization techniques to be used for learning linear discrimination weights in conjunction with sequence-level features. We compare our algorithm against state of the art kernel methods on a set of canonical datasets for structural and functional protein sequence classification.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 419–434, 2012.

2 Background

2.1 Support Vector Machines and Kernels

Support Vector Machines (SVMs) are linear classifiers; they assume that an input space, $\mathcal{X}$, can be partitioned by a hyperplane so that positive examples lie on one side of the plane and negative examples on the other. SVMs can capture nonlinear boundaries by mapping data into a transformed space, $\varphi : \mathcal{X} \to \mathcal{X}'$, where $\mathcal{X}$ is the original input space and $\mathcal{X}'$ is the transformed input space. Instead of computing this mapping directly, we can substitute the inner product between training examples in the transformed space, $\langle \varphi(x_i), \varphi(x_j) \rangle$, with a kernel function, $K(x_i, x_j)$, where $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $x_i, x_j \in \mathcal{X}$ [19].

String kernels extend the SVM to problem domains of variable-length sequences and also allow prior knowledge over a particular problem domain to be incorporated into the classifier. Examples of string kernels include the following: The *spectrum kernel* [9] computes, for a pair of sequences, x_i and x_j, the count of subsequences of length k that are present in both sequences. The *mismatch kernel* [10] can be best described as a fuzzy version of the spectrum kernel. For two sequences, x_i and x_j, the mismatch kernel computes the number of subsequences of length k across x_i and x_j that contain at most m mismatches. The *local alignment kernel* (LA-kernel)[17] computes the sum over all possible alignment scores between two sequences. Alignment scores generalize edit distance and score pairs of individual amino acids using a predefined distance matrix, commonly the BLOSUM62 matrix [4]. Profile kernels [15,7] are semi-supervised methods that augment training and test sequences with unlabeled sequences in the Protein Data Bank (PDB).

2.2 Hidden Markov Models

The Hidden Markov Model (HMM) [14] defines a probability distribution over sequences. The HMM assumes: (i) Each symbol in the sequence was generated from a mixture distribution; the mixture components are referred to as hidden states. (ii) The Markov property holds over hidden states i.e., the hidden state generating the current observation depends on the past only through the hidden state of the previous observation.

The joint probability of a sequence, $x_{1:T}$ of length T, and a set of hidden states, $z_{1:T}$, under an HMM is given as follows:

$$p(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p(z_t|z_{t-1})p(x_t|z_t) = \prod_{t=1}^{T} \theta^{trans}_{z_{t-1},z_t} \theta^{emit}_{x_t,z_t}, \tag{1}$$

where θ^{trans} is a set of transition probabilities and θ^{emit} is a set of emission probabilities. Detailed descriptions of parameters are given in Table 1. Converting transitions from adjacent hidden states over the length of the sequence to counts of emissions and transitions gives

$$p(x_{1:T}, z_{1:T}) = \prod_{k,k'} \left(\theta^{trans}_{k,k'}\right)^{n^{trans}_{k,k'}} \prod_{k,m} \left(\theta^{emit}_{k,m}\right)^{n^{emit}_{k,m}} \tag{2}$$

Table 1. Description of HMM parameters

K	the number of HMM hidden states
M	the number of possible amino acids
N	the number of protein sequences
T_n	the length of the n^{th} sequence
$\mathbf{x}_n$	is the n^{th} amino acid sequence, $x_{n,t}$ is the t^{th} symbol in the n^{th} sequence
y_n	indicates the category associated with the n^{th} sequence. $y_n \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of all categories
$\mathbf{z}_n$	is the n^{th} sequence of hidden states, $z_{n,t}$ is the value of the hidden state for the n^{th} sequence at position t. $\mathbf{z}_n \in \mathcal{Z}$, where $\mathcal{Z}$ the set of all possible hidden state sequences.
$\theta^{trans}_{k,k'}$	the probability of hidden state k occuring at position t when hidden state k' appears at position $t+1$
$w^{tran}_{k,k'}$	defined as $\log \theta^{trans}_{k,k'}$
$\theta^{emit}_{k,m}$	the probability of emitting symbol m at position t when hidden state at position t is k
$w^{emit}_{k,m}$	defined as $\log \theta^{emit}_{k,m}$
$\mathbf{w}$	defined as $\left[w^{trans}_{1,:} \dots w^{trans}_{K,:} w^{emit}_{1,:} \dots w^{emit}_{K,:}\right]^\top$, a vector comprising both the transition and emission weights - the subscripted ":" is matlab notation for the vector over the relevant index.
$n^{emit}_{k,m}$	the number of times hidden state k occurs in conjunction with observed symbol m
$n^{trans}_{k,k'}$	the number of times hidden state k occurs before hidden state k'

In the logarithm, the joint probability of a sequence and associated hidden states under the HMM is a linear function:

$$\begin{aligned}\log p(x_{1:T}, z_{1:T}) &= \sum_{k,k'} n^{trans}_{k,k'} \log \theta^{trans}_{k,k'} + \sum_{k,m} n^{emit}_{k,m} \log \theta^{trans}_{k,m} \\ &\stackrel{\text{def}}{=} \sum_{k,k'} n^{trans}_{k,k'} w^{trans}_{k,k'} + \sum_{k,m} n^{emit}_{k,m} w^{emit}_{k,m}\end{aligned} \tag{3}$$

where we define $w \stackrel{def}{=} \log \theta$ for both emissions and transitions. HMMs are probability distributions and thus require that $\sum_{\mathcal{X},\mathcal{Z}} p(\mathbf{x}, \mathbf{z}) = 1$, where $\mathcal{X}$ indicates the set of all possible sequences with alphabet size M, and $\mathcal{Z}$ indicates the set of all hidden states assignments for a sequence $x_{1:T}$. This constraint is satisfied as long as $\sum_{k'} \theta^{trans}_{k,k'} = 1$ and $\sum_m \theta^{emit}_{k,m} = 1$.

It is often useful to find the maximum probability assignment of values to hidden states, i.e., $\arg\max_{z_{1:T}} p(x_{1:T}, z_{1:T})$. The maximum can be computed efficiently by distributing the addition operator over max function to create the following recurrence:

$$\max_{z_{1:t}} \log p(x_{1:t}, z_{1:t}) = \max_{z_t} \left[\left(\max_{z_{1:t-1}} \log p(x_{1:t-1}, z_{1:t-1}) \right) + \log p(z_t | z_{t-1}) + \log p(x_t | z_t) \right] \tag{4}$$

The algorithm that uses this recurrence to compute the maximum over $z_{1:T}$ is known as the Viterbi algorithm [14].

2.3 Neural Networks

A feed-forward artificial neural network (ANN) is a nonlinear classifier or regression function where the input to output transformation is a composition of differentiable functions: $f^{(H)}(\dots(f^{(1)}(x))\dots)$. We assume a dataset $\{(x_n, y_n)\}_{n=1}^N$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$ where x_n is an input vector associated with an output y_n. We denote the vector valued output of each layer as $\mathbf{f}^{(h)}$, which we call "layer h" of the neural network. The top layer of the network (layer H) is compared against the true output, y, using a loss function $\ell(\mathbf{f}^{(H)}, y)$. In a standard ANN, $\mathbf{f}^{(h)}_i$, the i^{th} element of the vector $\mathbf{f}^{(h)}$, is computed by passing a linear combination of the values from the previous layer through a squashing function (usually the hyperbolic tangent function). Each $\mathbf{f}_i^{(h)}$ is computed independently of $\mathbf{f}_j^{(h)}$, $i \neq j$ given values from layer $h-1$:

$$\mathbf{f}^{(h)} = \left[f_1^{(h)} \left(\mathbf{f}^{(h-1)} \right), \ldots, f_{n_h}^{(h)} \left(\mathbf{f}^{(h-1)} \right) \right]^\top, \tag{5}$$

where n_h is the number of elements in layer h and $f_i^{(h)}$ denotes the composition of squashing and linear functions used to compute the i^{th} element of $\mathbf{f}^{(h)}$.

Convolutional Neural Networks (CNNs) [8] are inspired by neural connections in the human visual cortex. In CNNs, lower levels of the network respond to local portions of the input. For instance, the Time Delay Neural Network (TDNN) [20] is a type of CNN used in speech recognition. In the TDNN, the first hidden layer of the network is computed from a set of overlapping windows of an input sequence i.e., $\mathbf{f}^{(1)}$ is computed from an input sequence $x_{1:T}$:

$$\begin{aligned}\mathbf{f}^{(1)} = \Big[& f_1^{(1)} \left(x_{1:n_{cnv}} \right), f_1^{(1)} \left(x_{2:n_{cnv}+1} \right), \ldots, f_1^{(1)} \left(x_{T-n_{cnv}:T} \right), \ldots, \\ & f_{n_h}^{(1)} \left(x_{1:n_{cnv}} \right), f_{n_h}^{(1)} \left(x_{2:n_{cnv}+1} \right), \ldots, f_{n_h}^{(1)} \left(x_{T-n_{cnv}:T} \right) \Big]^\top \end{aligned} \tag{6}$$

where n_{h-1} indicates the number of elements in the vector $\mathbf{f}^{(h-1)}$ and n_{cnv} is the size of the "convolutional window" (the number of elements from the input, x, which contribute to produce the value of layer 1). We use Matlab slice notation, $x_{i_1:i_2}$, to indicate a sub-vector of the input sequence starting at index i_1 and ending at index i_2.

3 Sequence Classification with Subsequence Networks

Our family of feed-forward classification models are convolutional neural networks that assume a protein's structural or functional category can be predicted by the presence of a set of subsequences. We call these models Subsequence Networks (SN). In a Subsequence Network, the convolutional layer learns the degree to which a subsequence is present in a protein sequence. The degree of presence of a subsequence acts as a feature, which can be input to a linear classification layer, allowing combinations of these subsequence features to be detected.

Convolutional units in the Subsequence Network are structured like HMMs, except that we relax the constraint that the output of each unit defines a probability distribution over sequences. That is, we perform unconstrained optimization with respect to $w_{k,k'}^{trans} \overset{def}{=} \log \theta_{k,k'}^{trans}$ and $w_{k,m}^{emit} \overset{def}{=} \log \theta_{k,m}^{emit}$ rather than enforcing the constraint that each θ vector sums to one. We refer to these unnormalized models as "Sequence Scoring Models" (SSM). Figure 1 shows a diagram of our Subsequence Network using an SSM convolutional layer.

3.1 Pair-SSMs

The Pair-SSM is an unnormalized version of the Pair HMM [5]. Pair HMMs probabilistically extend the concept of edit distance by assigning probabilities

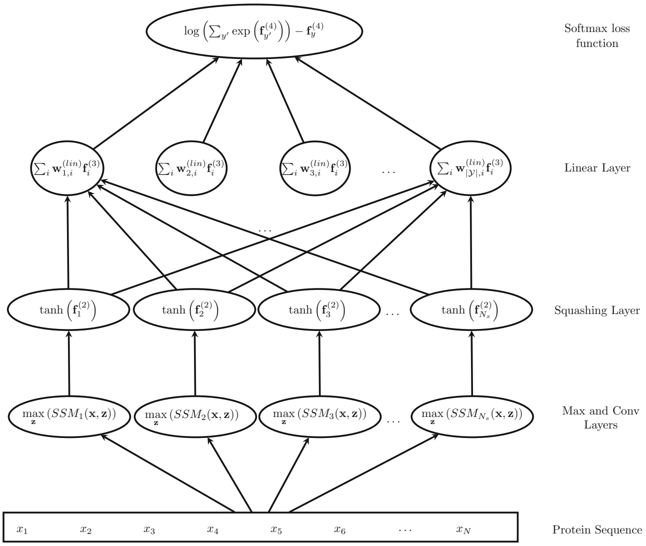

$\log\left(\sum_{y'} \exp\left(\mathbf{f}_{y'}^{(4)}\right)\right) - \mathbf{f}_y^{(4)}$
Softmax loss function
$\sum_i \mathbf{w}_{1,i}^{(lin)} \mathbf{f}_i^{(3)}$
$\sum_i \mathbf{w}_{2,i}^{(lin)} \mathbf{f}_i^{(3)}$
$\sum_i \mathbf{w}_{3,i}^{(lin)} \mathbf{f}_i^{(3)}$
...
$\sum_i \mathbf{w}_{|\mathcal{Y}|,i}^{(lin)} \mathbf{f}_i^{(3)}$
Linear Layer
...
$\tanh\left(\mathbf{f}_1^{(2)}\right)$
$\tanh\left(\mathbf{f}_2^{(2)}\right)$
$\tanh\left(\mathbf{f}_3^{(2)}\right)$
...
$\tanh\left(\mathbf{f}_{N_s}^{(2)}\right)$
Squashing Layer
$\max_{\mathbf{z}} (SSM_1(\mathbf{x}, \mathbf{z}))$
$\max_{\mathbf{z}} (SSM_2(\mathbf{x}, \mathbf{z}))$
$\max_{\mathbf{z}} (SSM_3(\mathbf{x}, \mathbf{z}))$
...
$\max_{\mathbf{z}} (SSM_{N_s}(\mathbf{x}, \mathbf{z}))$
Max and Conv Layers
x_1 x_2 x_3 x_4 x_5 x_6 ... x_N
Protein Sequence

Fig. 1. A diagram illustrating a Subsequence Network being applied to an input sequence. In the bottom row of the network, the maximum of the scores from each SSM are taken over the input sequence. The Conv layer is defined by a score from an SSM. In the second row, a squashing function is applied to the maximum SSM scores. The third row computes the distance of these squashed scores from hyperplanes used to define boundaries between sequence categories. Finally, the loss function compares the category given by the hyperplane to the true sequence category.

to a symmetric set of insertions, deletions, and substitutions that allows one sequence from a pair to be created from the other.

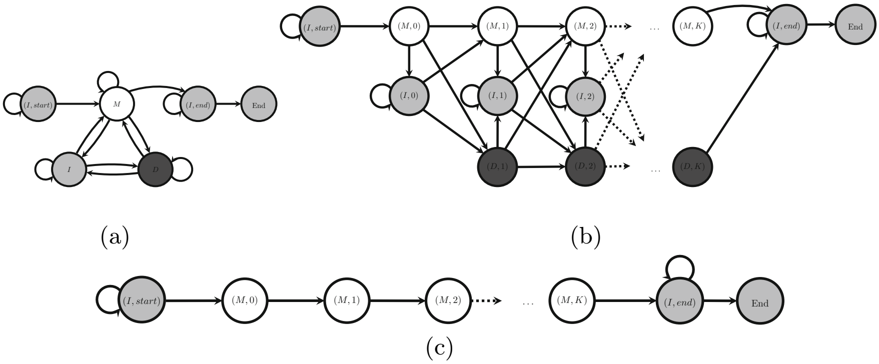
The log probability of a pair of sequences, $\mathbf{x}_i$ and $\mathbf{x}_j$, in the Pair HMM can be given by a linear model in the log of the distribution parameters:

$$\log p(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}) = \sum_{k,k'} n_{k,k'}^{trans} w_{k,k'}^{trans} + \sum_{m,m'} n_{m,m'}^{emit} w_{m,m'}^{emit} \tag{7}$$

where $n_{m,m'}^{emit}$ indicates the number of times we substituted an amino acid, m, from sequence $\mathbf{x}_i$ with m' from sequence $\mathbf{x}_j$, and $w_{m,m'}^{emit}$ is the associated cost of this substitution. The expression in Equation 7 differs from the probability of the standard HMM (Equation 3) in that we replace counts of emissions from a hidden state by counts of substitutions of amino acid m from sequence i with amino acid m' from sequence j i.e., $n_{k,m}^{emit}$ becomes $n_{m,m'}^{emit}$, and $w_{k,m}^{emit}$ becomes $w_{m,m'}^{emit}$. The Pair-SSM includes three types of hidden states: In an *Insert* hidden state, the model emits a symbol from sequence $\mathbf{x}_i$. In a *Delete* hidden state the model emits a symbol from sequence $\mathbf{x}_j$. In a *Match* hidden state, the model emits a symbol from both sequences. To allow the model to capture relevant subsequences, we add additional hidden states I_{start} and I_{end}. These hidden states emit symbols of either $\mathbf{x}_i$ or $\mathbf{x}_j$ from a background distribution, allowing the main portion of the Pair-SSM to emit symbols from the relevant subsequence. We show a diagram of hidden state transitions in Figure 2a.

If used in the convolutional layer of a Subsequence Network, the Pair-SSM extracts features using a similar technique as the LA-kernel. In this type of network, subsequences that best match an input sequence are selected from within each training set sequence by the Pair-SSMs. This view is closely related to the empirical kernel map [18,17]. In the empirical kernel map, a feature vector associated with an unknown sequence is given by a vector of kernel evaluations over the training set i.e., we map the sequence x to the vector $\left[K\left(x_1,x\right),\ldots,K\left(x_N,x\right)\right]^\top$, where $K\left(\cdot,\cdot\right)$ is the kernel function and $\left\{x_n\right\}, \; n \in [1\ldots N]$ is the set of training sequences. In the first layer of empirical kernel map, a feature vector is computed from an input sequence by evaluating a fixed kernel function on the input sequence paired with each training set sequence. This set of values are then combined linearly using weights, w_n, to produce an overall score for the query sequence: $s(x) = \sum_n w_n K(x_n, x)$. As in all linear classifiers, we classify the query sequence, x, as a member of the positive class if $s(x) > 0$ and as a member of the negative class otherwise. In SVM/kernel classification, kernel evaluations for all pairs of sequences are computed independently. Then, given $K(x_i, x_j)$, $\forall i, j$, the SVM learning algorithm solves a (convex) quadratic program to compute the linear weights, w_n.

Although optimization over the Subsequence Network with a Pair-SSM convolutional layer is tractable, it is not yet practical without distributing computation over multiple processors. For each SGD epoch we must compute the Viterbi paths over N^2 pairs of sequences, x_i and x_j, at a cost of $O(|x_i| \times |x_j|)$ (where N is the number of training sequences and $|x|$ is the length of a sequence).



(a) (b)

(c)

Fig. 2. A diagram of the deterministic finite-state automaton associated with **(a)** the Pair-SSM **(b)** the Local SSM (L-SSM) and **(c)** the Simplified Local SSM (SL-SSM). Match states are indicated with a white background, Insert states with a light-gray background, and Delete states with a dark-gray background.

3.2 Local SSM (L-SSM)

The Local SSM (Figure 2b) is an unnormalized version of the Profile HMM (pHMM) adapted to model a single subsequence within an observed sequence. Profile HMMs [6] are a variation of the standard HMM commonly used for modeling biological sequences. They are left-to-right, non-ergodic HMMs [14] that represent sequences in relation to an archetypal sequence encoded in the emission distributions of the pHMM's hidden states. Profile HMMs use three types of hidden states: (1) *Match (M)* states encode individual symbols of the archetypal sequence, (2) *Insert (I)* states allow additional symbols to be inserted between matched symbols, and (3) *Delete (D)* states allow matched symbols to be skipped. Hidden states of the archetypal sequence are expressed as pairs of symbols (s,k), where $s \in \{M, I, D\}$ indicates a Match (M), Insert (I), or Delete (D), paired with a base state, $k \in [1 \ldots K]$, which can be thought of as indexing a symbol in the archetypal sequence. The form of the archetypal sequence is encoded by the emission distributions from each of the K match states. In the local version of the pHMM, which models a single subsequence within the observed sequence, I_{start} and I_{end} are special insert states that allow portions of the sequence before the archetypal sequence to be skipped. We fix transition and emission probabilities from I_{start} and I_{end}, allowing these to be ignored during optimization. In addition, we explicitly include an *End* state to mark the end of the observed sequence. We must include the *End* state because without transition from I_{end} to *End*, the model favors archetypal subsequences positioned near the beginning of the observed sequence. As in the standard pHMM, in our L-SSM, emissions occur only from Match and Insert states.

3.3 Simplified Local SSM (SL-SSM)

Like the L-SSM, the SL-SSM models relevant subsequences within a set of sequences. The SL-SSM (Figure 2c) simplifies the L-SSM by removing Insert and Delete states from the model. This change in the model results in contiguous subsequences of match states. This structural change speeds inference i.e., the highest scoring set of hidden states can be efficiently computed by sliding the window of K match states over a sequence and returning the position with the highest probability. We use the same parametrization of hidden states as in the L-SSM: $\{I_{start}$,$(M,1)$,...,(M,K),I_{end},END$\}$, where the pair (M,k), indicates a hidden state that emits the k^{th} symbol of the archetypal sequence. Transitions from the Match state (M,k) , $k < K$ to $(M, k+1)$ and transitions from the state (M,K) to the state I_{end} occur with probability one.

The L-SSM or SL-SSM convolutional layer in the Subsequence Network can be interpreted as a simplification of the SN with a Pair-SSM convolutional layer. This simplification is motivated by two assumptions about the domain of protein sequences: (i) the set of protein sequences lies on a lower-dimensional manifold within the sequence space and (ii) the basis given by the training set spans the manifold of our domain and is redundant. With these assumptions it becomes reasonable to simplify the Pair-SSM convolutional layer by creating a model with a lower-dimensionality basis independent of the training examples. For our model, we choose this basis to be a fixed set of L-SSMs or SL-SSMs.

When we replace Pair-SSMs with (S)L-SSMs, additional computational efficiencies become possible because the (S)L-SSM allows us to store only the locally relevant pattern rather than an entire sequence. Computing the score of a sequence under an (S)L-SSMs where hidden states are restricted to small, fixed lengths therefore requires less computation time than evaluating the Pair-SSM between pairs of sequences. A disadvantage of these simplifications is that new parameters are added to the model. In particular, the number of (S)L-SSMs and the number of hidden states for each (S)L-SSM must be specified in advance. In the results section, we show that these simplifications not only maintain much of the accuracy of the Pair-SSM layer, but they are also robust to variations in parameter choices.

3.4 Subsequence Network Objective Function

The objective function for our model includes a loss for each sequence in the dataset and a regularization term that penalizes large parameter values:

$$F(\mathbf{x}) = \sum_n \ell\left(\mathbf{x}_n, y_n; \mathbf{w}^{all}\right) + \frac{\lambda}{2}||\mathbf{w}^{all}||^2 \tag{8}$$

where $\mathbf{w}^{all}$ is an agglomeration of all of the SSM weight vectors in the model (the composition of $\mathbf{w}^{all}$ varies depending on which type of SSM is used for the convolutional layer) and linear combination weights associated with each SSM; $\ell(x, y; \mathbf{w}^{all})$ is a loss function. The λ term determines the trade off between the loss and magnitude of the weights.

A loss function compares the output of a CNN with the label of a single protein sequence. In our model, we use the a softmax loss, shown in the first row of Table 3. The output of the network is given by

$$\mathbf{f}^{(4)} \overset{def}{=} f^{(4)}\left(f^{(3)}\left(f^{(2)}\left(f^{(1)}\left(\mathbf{x}_n\right)\right)\right)\right) \overset{def}{=} \text{Lin}\left(\text{Tanh}\left(\text{Max}\left(\text{Conv}\left(\mathbf{x}_n\right)\right)\right)\right) \tag{9}$$

where *Conv* is a convolutional layer containing multiple convolutional units, *Max* computes the maximum over the responses of each convolutional unit, *Tanh* is the hyperbolic tangent function and maps values in the range $(-\infty, \infty)$ to $(-1, 1)$, *Lin* is a linear layer. As in Section 2.3, we denote the output from hidden layer h as $\mathbf{f}^{(h)}$. Table 3 gives the full form of each layer of the network, and Table 2 gives a description of network parameters.

If the number of hidden states for each SSM, $|\mathcal{Z}^{(i)}|$, across an individual network is the same $\left(|\mathcal{Z}^{(i)}| = |\mathcal{Z}^{(j)}| \quad \forall i, j\right)$ layer $\mathbf{f}^{(1)}$ becomes a matrix of size $N_s \times |\mathcal{Z}|$, where N_s is the number of SSMs in the convolutional layer. In the L-SSM and

Table 2. Subsequence Network parameters

Parameter	Definition
N_s	number of SSMs in the convolutional layer
$\mathbf{f}^{(h)}$	the vector of values that comprise hidden layer h
$\mathbf{w}^{all}$	a combined vector of all SSM and linear weights, $\left[\left(\mathbf{w}^{(lin)}\right)^\top, \left(\mathbf{w}^{(1)}\right)^\top, \ldots, \left(\mathbf{w}^{(N_s)}\right)^\top\right]^\top$
$\mathbf{w}^{(lin)}$	linear weights that define linear boundaries between dataset categories, $\mathbf{w}^{(lin)} = \left[w^{(lin)}_{1,1}, \ldots, w^{(lin)}_{1,N_s}, \ldots, w^{(lin)}_{\|\mathcal{Y}\|,1}, \ldots, w^{(lin)}_{\|\mathcal{Y}\|,N_s}\right]$
$\mathbf{w}^{(i)}$	a vector of transition and emission weights for the i^{th} SSM, $\mathbf{w}^{(i)} = \left[\left(w^{trans}_{1,:}\right)^{(i)}, \ldots, \left(w^{trans}_{K,:}\right)^{(i)}, \left(w^{emit}_{1,:}\right)^{(i)}, \ldots, \left(w^{emit}_{K,:}\right)^{(i)}\right]^\top$ where K is the number of hidden states in the SSM
$\mathcal{Z}^{(i)}$	the set of hidden states for the i^{th} SSM

Table 3. The table above describes the composition of each layer in the Subsequence Network and gives an expression for the Jacobian with respect to the layer's input. The values of each layer are given by the vector $\mathbf{f}^{(h)}$ for hidden layer h. The Jacobian of the first layer (Conv) with respect to the input is not used during inference.

Layer	Definition	Jacobian with respect to $\mathbf{f}^{(h)}$
$\ell\left(\mathbf{f}^{(4)}, y\right)$	$\log \sum_i \exp\left(\mathbf{f}_i^{(4)}\right) - \mathbf{f}_y^{(4)}$	$\begin{bmatrix} \frac{\exp\left(\mathbf{f}_1^{(4)}\right)}{\sum_i \exp\left(\mathbf{f}_i^{(4)}\right)}, \ldots, \frac{\exp\left(\mathbf{f}_{\lvert\mathcal{Y}\rvert}^{(4)}\right)}{\sum_i \exp\left(\mathbf{f}_i^{(4)}\right)} \\ \vdots \\ \frac{\exp\left(\mathbf{f}_1^{(4)}\right)}{\sum_i \exp\left(\mathbf{f}_i^{(4)}\right)}, \ldots, \frac{\exp\left(\mathbf{f}_{\lvert\mathcal{Y}\rvert}^{(4)}\right)}{\sum_i \exp\left(\mathbf{f}_i^{(4)}\right)} \end{bmatrix} - I$
$\mathbf{f}^{(4)} \equiv \mathrm{Lin}\left(\mathbf{f}^{(3)}\right)$	$\left[\sum_{i=1}^{N_s} w_{1,i}^{(lin)} \mathbf{f}_i^{(2)}, \ldots, \sum_{i=1}^{N_s} w_{\lvert\mathcal{Y}\rvert,i}^{(lin)} \mathbf{f}_i^{(2)}\right]^\top$	$\begin{bmatrix} w_{1,1}^{(lin)}, \ldots, w_{\lvert\mathcal{Y}\rvert,1}^{(lin)} \\ \vdots \\ w_{1,N_s}^{(lin)}, \ldots, w_{\lvert\mathcal{Y}\rvert,N_s}^{(lin)} \end{bmatrix}$
$\mathbf{f}^{(3)} \equiv \mathrm{Tanh}(\mathbf{f}^{(2)})$	$\left[\tanh\left(\mathbf{f}_1^{(2)}\right), \ldots, \tanh\left(\mathbf{f}_{N_s}^{(2)}\right)\right]^\top$	$\left[d\tanh\left(\mathbf{f}_1^{(2)}\right), \ldots, d\tanh\left(\mathbf{f}_{N_s}^{(2)}\right)\right]^\top I$
$\mathbf{f}^{(2)} \equiv \mathrm{Max}\left(\mathbf{f}^{(1)}\right)$	$\begin{bmatrix} \max\left(\left[f_{1,1}^{(1)}, \ldots, f_{1,\lvert\mathbf{Z}^{(1)}\rvert}^{(1)}\right]\right) \\ \vdots \\ \max\left(\left[f_{N_s,1}^{(1)}, \ldots, f_{N_s,\lvert\mathbf{Z}^{(N_s)}\rvert}^{(1)}\right]\right) \end{bmatrix}$	$\begin{bmatrix} \mathbb{I}\left(f_{1,1}^{(1)} = \max\left(f_{1,:}^{(1)}\right)\right), \ldots, \mathbb{I}\left(f_{N_s,1}^{(1)} = \max\left(f_{N_s,:}^{(1)}\right)\right) \\ \vdots \\ \mathbb{I}\left(f_{1,\lvert\mathcal{Z}^{(1)}\rvert}^{(1)} = \max\left(f_{1,:}^{(1)}\right)\right), \ldots, \mathbb{I}\left(f_{N_s,\lvert\mathcal{Z}^{(1)}\rvert}^{(1)} = \max\left(f_{N_s,:}^{(1)}\right)\right) \end{bmatrix}$
$\mathbf{f}^{(1)} \equiv \mathrm{Conv}(x)$	$\begin{bmatrix} \mathrm{SSM}_1\left(x, \mathbf{z}_1^{(1)}\right), \ldots, \mathrm{SSM}_1\left(x, \mathbf{z}_{\lvert\mathcal{Z}^{(1)}\rvert}^{(1)}\right) \\ \vdots \\ \mathrm{SSM}_{N_s}\left(x, \mathbf{z}_1^{(N_s)}\right), \ldots, \mathrm{SSM}_{N_s}\left(x, \mathbf{z}_{\lvert\mathcal{Z}^{(N_s)}\rvert}^{(N_s)}\right) \end{bmatrix}$	

Pair-SSMs, $|\mathcal{Z}|$ is exponential in the size of the input sequence. We make the feed-forward and backpropagation steps for the network tractable by computing the composition Max(Conv(x)) directly using the Viterbi algorithm [14]. The locations of the non-zero indicator functions in the Jacobian of the Max layer are then given by the Viterbi path [14] through the SSM.

Our Subsequence Network incorporates the hyperbolic tangent squashing function, $\tanh(x) = \frac{e^{-2x}+1}{e^{-2x}-1}$. We denote the derivative of this function with respect to the hyperbolic tangent input as $d\tanh(x) = 1 - \tanh^2(x)$. In the Max layer, the function $\max(\boldsymbol{v})$ returns the largest scalar element of the vector $\boldsymbol{v}$.

3.5 Training Subsequence Networks

Training is performed using the stochastic gradient descent (SGD) algorithm. In SGD, the gradient of the objective is evaluated for each training example. The gradient is then scaled by a learning rate and subtracted from the current set of parameters to obtain a new set of parameters. This procedure contrasts with batch gradient learning where the gradient is computed for the entire set of training examples. We compute gradients using the backpropagation procedure [16]. Our model includes a locally non-smooth Max function, causing the the gradient of the objective to be undefined at the non-smooth points. To deal with this potential issue, we skip the gradient update in these non-smooth areas [3].

SGD updates take the form

$$\mathbf{w}_t^{all} \leftarrow \mathbf{w}_{t-1}^{all} - \eta_t \frac{\partial F(\mathbf{x}_n)}{\partial \mathbf{w}^{all}} \tag{10}$$

where $F(\mathbf{x}_n) = \ell\left(\mathbf{x}_n, y_n; \mathbf{w}^{all}\right) + \frac{\lambda}{2}\left|\left|\mathbf{w}^{all}\right|\right|^2$ is the objective for a single sequence, t indicates the iteration number in the SGD algorithm, $\mathbf{w}_t^{all}$ indicates the value of the weights at the t^{th} iteration, and η_t is the learning rate at iteration t and has the form $\eta_t = \eta_0 (1 + \lambda\eta_0 t)^{-1}$, where λ is the regularization parameter.

The gradient with respect to the linear weights is given by

$$\frac{\partial F(x_n)}{\partial w_{yi}^{(lin)}} = -\frac{\partial \ell\left(\mathbf{x}_n, y_n; \mathbf{w}^{all}\right)}{\partial \mathbf{f}_y^{(4)}} \mathbf{f}_i^{(3)} \tag{11}$$

This leads to an update where w_{yi} (the linear weight associated with the i^{th} SSM and category y) is increased if the current training example matches the weight's category and decreased otherwise. The change in the weight's value is proportional to, $\mathbf{f}_i^{(3)}$, the squashed response of the i^{th} SSM. The expression for $\frac{\partial \ell\left(\mathbf{x}_n, y_n; \mathbf{w}^{all}\right)}{\partial \mathbf{f}_y^{(4)}}$ is given in Table 3.

The gradient with respect to the SSM weights is given by

$$\frac{\partial F(x_n)}{\partial \mathbf{w}^{(i)}} = \frac{\partial \ell\left(x_n, y_n\right)}{\partial \mathbf{f}_i^{(3)}} \frac{\partial \mathbf{f}_i^{(3)}}{\partial \mathbf{w}^{(i)}} \tag{12}$$

where

$$\frac{\partial \ell}{\partial \mathbf{f}_i^{(3)}} = \left(\sum_{y \in \mathcal{Y}} \frac{\exp\left(\sum_{i=1}^{N_s} w_{yi}^{(lin)} \mathbf{f}_i^{(3)}\right) w_{yi}^{(lin)}}{\sum_{y' \in \mathcal{Y}} \exp\left(\sum_{i=1}^{N_s} w_{y'i}^{(lin)} \mathbf{f}_i^{(3)}\right)} - w_{y_n i}^{(lin)} \right) \tag{13}$$

The gradient of $\mathbf{f}_i^{(3)}$ with respect to the SSM weights, $\mathbf{w}^{(i)}$, is given by $\frac{\partial \mathbf{f}_i^{(3)}}{\partial \mathbf{w}^{(i)}} = d\tanh\left(\mathrm{SSM}_i\left(x_n\right)\right) \mathbf{n}_{\mathbf{z}_{max}}^{(i)}$ where $\mathbf{n}_{z_{max}}^{(i)}$ is a vector of counts of emissions and transitions associated with the set of hidden states that maximizes the value of $\mathrm{SSM}_i\left(\mathbf{x}_n, \mathbf{z}\right)$ and $d\tanh$ is the derivative of the tanh function with respect to its input. As in the HMM, $\mathbf{z}_{max}$ for the SSM can be computed efficiently with the Viterbi algorithm [14]. Gradient steps therefore change $\mathbf{w}^{(i)}$ in proportion to the counts of emissions and transitions in the highest scoring set of hidden states. The factor of proportionality is $\frac{\partial \ell}{\partial \mathbf{f}_i^{(3)}}$. This factor can be viewed as a measure of how much the i^{th} linear weight of the ground truth class, $w_{y_n i}^{(lin)}$, differs from its expected value.

4 Experiments

We perform classification experiments on four protein datasets. Of these datasets, two are derived from the Structural Classification of Proteins (SCOP) [12] version 1.53. SCOP is a database that categorizes proteins with known structure into a hierarchy with levels denoted by class, fold, superfamily, and family, from broadest level to the most narrow respectively. The first structural dataset [11], denoted by SF, defines 54 fixed superfamily partitions. The second dataset [15], FD, consists of 23 predefined partitions at the fold level. Both of the SCOP datasets were constructed so that no overlap between lower levels in the hierarchy occurs between training and test sets.

The other protein datasets divide sequences into functional, rather than structural, categories. The enzyme classification dataset [13], which we refer to as EC, contains sequences from six enzyme categories and a set of non-enzymes for a total of 7 one-versus-rest datasets. The fourth dataset [13] categorizes proteins

Table 4. Datasets Sizes - # Train indicates the average number of sequences in the training set over all categories, # Test indicates the average number of test set sequences, and # Categories indicates the number of one-versus-rest classification problems defined by the dataset

Dataset	# Train	# Test	# Categories
SF	2948	1366	54
FD	2196	2155	23
EC	379	110	7
GO	115	57	23

by Gene Ontology. We refer to this dataset as GO. Information about the protein datasets is given in Table 4.

4.1 Comparative Classifiers

We compared the three SVM string kernels to our Subsequence Network. The BLAST kernel was computed by performing a BLAST [1] database search on each sequence. If another sequence from the training set was returned by the BLAST search, then we set the corresponding Kernel value to the returned E-value. The mismatch kernel was described in Section 2.1 and has two parameters. We denote a mismatch evaluation by Mismatch(k, m), where k is the subsequence length and m is the number of allowable mismatches. The LA-Kernel was also described in Section 2.1. For all experiments, the LA-Kernel's temperature parameter, β, was set to 0.2.

4.2 Models and Parameters

We compared three variations of our Subsequence Network. In the first variation, "Pair-SSM," the convolutional layer consisted of Pair-SSMs associated with each training sequence in the model. Similarly, the "L-SSM" and "SL-SSM" variations use L-SSMs and SL-SSMs in the convolutional layer respectively.

For the Pair-SSM network, we initialized pairwise weights using a scaled version of the BLOSUM62 matrix [4] and ran inference for 5 epochs on the FD dataset and 10 epochs on the EC and GO datasets. We set the precision parameter associated with the Gaussian regularizer to $\lambda = .005$. We set multiplicative factor in the learning rate (Section 3.5) $\eta_0 = .1$ for the linear weights. For the Pair-SSM parameters, we set $\eta_0 = \frac{.1}{10 \times (\#\text{Train})}$, where # Train is the number of training set sequences. To allow training of the Pair-SSM model to take place in a reasonable amount of time, we distributed gradient computations of SSMs in the convolutional layer within each backpropagation step over 50 machines[1].

Choices of parameters were the same for the L-SSM and SL-SSM networks: We used 96 SSMs in the convolutional layer and set K (the number of states

[1] Training the (S)L-SSM networks was significantly faster than the Pair-SSM network. To give a rough comparison, SL-SSM network training with the parameters described was faster than computing the Mismatch(5,2) kernel.

for each SSM) to 11. We set the precision parameter to $\lambda = .005$ for both SSM weights and linear weights, and we set $\eta_0 = .1$. For each experiment, we ran inference for 30 epochs.

The weight vector for these models were set by generating subsequences, x, of length K uniformly and at random. For position k in the subsequence, weight w_{kx_k} was set to $\frac{1}{K}$ and weights w_{km}, $m \neq x_k$ was set to $-\frac{1}{K}$.

To compensate for unbalanced numbers of positive and negative examples, we oversampled the positive training set so that the same number of positive and negative examples were presented to the SGD trainer during each epoch. In the (S)L-SSM networks, we found that initializing the linear weights so that half of the SSMs were associated with the positive class and the other half associated with the negative class improved performance of our algorithm.

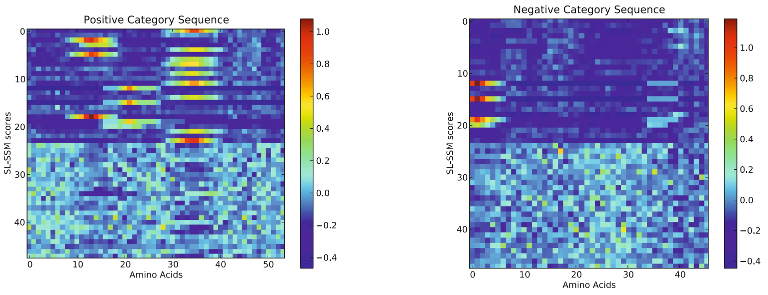
4.3 Evaluation Metrics

We measured the performance of our algorithm by computing the average ROC scores for each of the one-versus-rest classification problems defined by our datasets. ROC, also known as Area Under the Curve (AUC), is defined as the area under the receiver operating characteristic (ROC) curve. The ROC curve plots the percentage of true positives against the percentage of false positives.We also report ROC$_{50}$ and ROC$_{10\%}$ which are the area under the ROC curve excluding all but the top 50 negative examples or 10 percent of the negative examples respectively. To compare the performance of different models, we performed the Wilcoxson signed rank tests at a 5% significance level using each one-versus-rest category. We report ROC results based on the algorithm's scoring of sequences on the test sets.

4.4 Synthetic Experiments

We constructed a synthetic dataset to verify that our network can detect the relevant subsequence features that we propose will lead to good protein sequence classification performance. Specifically, we generated 1000 sequences with lengths generated from a Poisson distribution with a mean of 50 symbols. Each sequence contained between one and three fixed relevant subsequences, with the positive class containing all three subsequences and the negative class containing either one or two sequences of any type. The relevant subsequences were arranged in random order within the sequence. After placement of the relevant sequences, noise was added - we replaced each every relevant subsequence amino acid with a random amino acid with 10% probability. Amino acids outside relevant subsequences were generated from a uniform multinomial distribution.

Figure 3 shows responses of the lowest layer in the SL-SSM network on two example sequences from the synthetic dataset. Strong responses in portions of the sequences that contain relevant subsequences indicate that our model is able to effectively learn features that discriminate well for a dataset where categories are determined based on the presence of subsequences. The SL-SSM achieves an ROC of .9998 and ROC$_{50}$ of .997 on the test portion of the synthetic dataset, showing that after detecting relevant subsequences, our model has the ability



(a) (b)

Fig. 3. The figures above show responses from each of the 48 unnormalized SL-SSMs over each length 7 subsequence for a sequence generated from the positive class (a) and the negative class (b). The positive example contains all three relevant subsequences while the negative example contains only one relevant sequence. The first 24 SL-SSMs (top half both figures) were constrained to be associated with the positive category, while the last 24 were constrained to be associated with the negative category. The heat maps show that sets of positive SL-SSMs have adapted to each of the three relevant subsequences in the synthetic dataset - both the three relevant subsequences in the positive example and the one relevant subsequence in the negative example were detected by subsets of the first 24 SL-SSMs. In contrast, SL-SSMs associated with the negative category learn a background distributions of symbols.

to effectively classify sequences generated according to a relevant subsequences assumption on the dataset.

4.5 Parameter Adjustment

We adjusted the parameters of (S)L-SSM models by comparing both the number of SSMs in each network and the number of hidden states for the SSMs (we used the same number of hidden states for all SSMs) in experiments on the FD dataset. Table 5 shows results from these comparisons. Within a relatively wide range

Table 5. Average ROC results for different settings of the SL-SSM network on the FD dataset. ROCs were averaged over ten independent trials initialized with random pattern weights. When varying the number of SL-SSM hidden states in (a), 96 SL-SSMs were used in the network. In (b), 11 hidden states were used for each SL-SSM when varying the number of SL-SSMs.

Hidden States	ROC	ROC$_{50}$
7	.801	.146
9	.813	.145
11	.815	.153
13	.815	.153
15	.807	.163

(a)

# SL-SSMs	ROC	ROC$_{50}$
64	.812	.144
80	.812	.156
96	.814	.153
112	.816	.145
128	.816	.147

(b)

of parameter settings, the performance of the model stays roughly the same, showing that, although our method may require some adjustment using cross validation, micromanagement of parameter settings is not critical to maintaining acceptable performance. For the results shown in Table 6, we selected parameter settings for the other experiments ($K = 11$ with 96 convolutional units) that performed best on FD.

4.6 Protein Classification Experiments

We compared four subsequence models used in the convolutional layer of our Subsequence Network. The lower rows of Table 6a and b show results for the L-SSM and the SL-SSM. These indicate that the simpler model, the SL-SSM outperforms the L-SSM. Superior performance of the SL-SSM results from unstable inference in the L-SSM when attempting to learn insert transition weights. If these weights grow above zero for the positive class, then the model tends to explain every protein sequence using long sequences of insert states. For this reason, we fix the insert transition weights in the L-SSM to small negative values ($-\frac{1}{2K}$ where K is the number of Match states in the L-SSM), but this fix affects the flexibility of the model. On the FD dataset, the Pair-SSM outperforms our other models in both ROC and ROC$_{50}$. However, on the functional datasets, both L-SSM and SL-SSM outperform the Pair-SSM. These results indicate that the simpler (S)L-SSM assumptions may be better models of protein structure in certain cases.

Table 6 also compares our subsequence networks to SVM/kernel methods from the literature. Compared to the LA-kernel on the FD dataset, the Pair-SSM model is statistically equivalent in both ROC and ROC$_{50}$ measurements. Compared to the Mismatch kernel on the FD dataset, the Pair-SSM model performs better in ROC but is equivalent for ROC$_{50}$. We note, however, that the β parameter used for the LA-kernel is the best of many settings on both the SF and FD training/test set split. Due to this extensive adjustment on the FD dataset, it is likely that the LA-kernel overfits. In contrast, Pair-SSM network performance is only weakly dependent on parameter settings (the regularization parameter) and we did not perform extensive adjustment of these values, so overfitting likely to be less problematic for our Pair-SSM on the FD test set. The Pair-SSM is equivalent to both the LA-kernel and Mismatch kernel on the EC dataset in both ROC and ROC$_{10\%}$. On the GO dataset, the the Pair-SSM is outperformed by the LA-kernel in both ROC and ROC$_{10\%}$ but is equivalent to the Mismatch kernel.

In ROC, the LA-kernel outperforms the L-SSM and SL-SSM models on the SF and GO datasets but is statistically equivalent for the FD and EC datasets. In ROC$_{50}$, the LA-kernel outperforms the L-SSM and SL-SSM on both the SF and FD datasets at a 5% significance level. In ROC$_{10\%}$, both of our methods outperform the LA-kernel on the EC dataset. On the GO dataset, the LA-kernel's performance is statistically equivalent to the SL-SSM but outperforms the L-SSM. Compared to the Mismatch(5,2) kernel, both the L-SSM and SL-SSM models have equivalent performance in SF and FD in both ROC and ROC$_{50}$. Our algorithms outperform the Mismatch kernel in ROC$_{10\%}$ on both of the functional datasets.

Table 6. ROC results for the FD and SF datasets (a) and the EC and GO datasets (b). Because our model is non-convex, we report means and standard deviations of ROCs from multiple starting points in the SSM weight space. Ten trials were averaged for both the L-SSM and SL-SSM models for both structural and functional datasets. Due to the length of P-SSM network's runtime, we report results from only a single trial.

Dataset	SF		FD	
Method	ROC	ROC$_{50}$	ROC	ROC$_{50}$
BLAST	.756	.341	.603	.100
Mismatch (5,1)	.872	.403	.802	.146
Mismatch (5,2)	.888	.479	.782	.188
LA-Kernel($\beta = .2$)	.926	.663	.805	.216
Pair-SSM	-	-	0.826	0.168
L-SSM	0.901 ± 0.003	0.456 ± 0.013	0.804 ± 0.006	0.141 ± 0.010
SL-SSM	0.903 ± 0.003	0.503 ± 0.012	0.815 ± 0.009	0.153 ± 0.008

(a)

Dataset	EC		GO	
Method	ROC	ROC$_{10\%}$	ROC	ROC$_{10\%}$
Mismatch (4,1)	.580	.007	.740	.238
Mismatch (5,2)	.581	.008	.747	.245
LA-Kernel($\beta = .2$)	.574	.004	.801	.339
Pair-SSM	.544	.021	.737	.264
L-SSM	0.587 ± 0.018	0.081 ± 0.011	0.736 ± 0.004	0.280 ± 0.010
SL-SSM	0.583 ± 0.017	0.092 ± 0.024	0.731 ± 0.011	0.306 ± 0.018

(b)

For the EC dataset, none of the algorithms perform particularly well. It is possible that both the size of the dataset and weak correlations between sequence and function cause subsequence-based approaches to fail. A class of methods based on a different set of assumptions may be necessary to achieve strong functional classification performance.

5 Conclusions

The empirical kernel map applied in conjunction with SVM classifiers is strongly related to feed-forward models like convolutional neural networks. Based on this relationship, we show how to construct a family of models, which we call Subsequence Networks, where kernel parameters can be learned in conjunction with linear classification boundaries. Our Subsequence Networks operate differently from state-of-the-art protein sequence classification models yet can achieve comparable performance. We hope that Subsequence Networks can shift the focus in biological sequence classification from increasingly fine-tuned kernel methods toward developing structures with self-tuning abilities.

Our networks also contribute to existing neural network literature by extending the convolutional layer to a maximization over latent parameter spaces in standard sequence models. The effectiveness of this framework for protein sequence classification shows that it has potential in other classification domains.

A straightforward and potentially useful modification to our networks involves adapting them to take Psi-BLAST profiles, rather than sequences, as input. This shift to a semi-supervised structure has the potential to improve classification performance to the same degree as it has for kernel methods [15].

References

1. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic acids research 25(17), 3389–3402 (1997)
2. Blasiak, S., Rangwala, H.: A hidden markov model variant for sequence classification. In: International Joint Conference on Artificial Intelligence (2011)
3. Bottou, L.: Online algorithms and stochastic approximations. Online Learning and Neural Networks (1998)
4. Durbin, R.: Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge Univ. Pr. (1998)
5. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological sequence analysis. Cambridge University Press, Cambridge (2002)
6. Eddy, S.R.: Profile hidden markov models. Bioinformatics 14(9), 755 (1998)
7. Kuang, R., Ie, E., Wang, K., Siddiqi, M., Freund, Y., Leslie, C.: Profile-based string kernels for remote homology detection and motif extraction. In: 2004 IEEE Proceedings of Computational Systems Bioinformatics Conference, CSB 2004, pp. 152–160 (2004)
8. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
9. Leslie, C., Eskin, E., Noble, W.S.: The spectrum kernel: a string kernel for svm protein classification. In: Pacific Symposium on Biocomputing, Hawaii, USA, vol. 575, pp. 564–575 (2002)
10. Leslie, C.S., Eskin, E., Cohen, A., Weston, J., Noble, W.S.: Mismatch string kernels for discriminative protein classification. Bioinformatics 20(4), 467 (2004)
11. Liao, L., Noble, W.S.: Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. Journal of computational biology 10(6), 857–868 (2003)
12. Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. Journal of molecular biology 247(4), 536–540 (1995)
13. Qiu, J., Hue, M., Ben-Hur, A., Vert, J.P., Noble, W.S.: A structural alignment kernel for protein structures. Bioinformatics 23(9), 1090–1098 (2007)
14. Rabiner, L., Juang, B.: An introduction to hidden markov models. IEEE ASSP Magazine 3(1), 4–16 (1986)
15. Rangwala, H., Karypis, G.: Profile-based direct kernels for remote homology detection and fold recognition. Bioinformatics 21(23), 4239 (2005)
16. Rumelhart, D.E., Hintont, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
17. Saigo, H., Vert, J.P., Ueda, N., Akutsu, T.: Protein homology detection using string alignment kernels. Bioinformatics 20(11), 1682–1689 (2004)
18. Schölkopf, B., Mika, S., Burges, C.J.C., Knirsch, P., Müller, K.R., Rätsch, G., Smola, A.J.: Input space versus feature space in kernel-based methods. IEEE Transactions on Neural Networks 10(5), 1000–1017 (1999)
19. Smola, A.J., Schölkopf, B.: Learning with kernels. Citeseer (1998)
20. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.J.: Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech and Signal Processing 37(3), 328–339 (1989)

General Algorithms for Mining Closed Flexible Patterns under Various Equivalence Relations

Tomohiro I, Yuki Enokuma, Hideo Bannai, and Masayuki Takeda

Department of Informatics, Kyushu University, Fukuoka 819-0395, Japan
{tomohiro.i,bannai,takeda}@inf.kyushu-u.ac.jp,
yuki.enokuma@i.kyushu-u.ac.jp

Abstract. We address the closed pattern discovery problem in sequential databases for the class of *flexible* patterns. We propose two techniques of coarsening existing equivalence relations on the set of patterns to obtain new equivalence relations. Our new algorithm GenCloFlex is a generalization of MaxFlex proposed by Arimura and Uno (2007) that was designed for a particular equivalence relation. GenCloFlex can cope with existing, as well as new equivalence relations, and we investigate the computational complexities of the algorithm for respective equivalence relations. Then, we present an improved algorithm GenCloFlex+ based on new pruning techniques, which improve the delay time per output for some of the equivalence relations. By computational experiments on synthetic data, we show that most of the redundancies in the mined patterns are removed using the proposed equivalence relations.

1 Introduction

Discovering frequent patterns in sequence databases has great importance in a wide-range of areas, including analysis of customer purchasing histories, Web click streams, DNA/RNA sequences, natural language texts, and so on. Recent decades have seen the series of studies; Agrawal and Srikant [1] was one of the pioneering works on sequential pattern mining, and many studies followed [3,11,14].

In practical applications of pattern mining, a typical tradeoff to be considered is: On one hand, we would like to consider for the mining task, a rich set of patterns and a relatively low minimum support threshold so that we may discover interesting, possibly subtle information buried in the data. On the other hand, by choosing such a search space, a mining algorithm may give us a tremendous number of patterns as output, which will definitely be a bottle neck when the results are examined by domain experts. To deal with this problem, an important technique in reducing the number of patterns output without sacrificing their diversity, is to introduce an appropriate equivalence relation $\equiv$ on the pattern set Π, and to output only *closed* patterns, where a pattern P is closed if it is maximal in the equivalence class $[P]_\equiv$ to which P belongs under $\equiv$. This problem is referred to as *closed pattern discovery* and has been studied extensively [2,5–8,10,12,13,15,16,19–21].

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 435–450, 2012.

In this paper, we consider the closed pattern discovery problem for the class of flexible patterns. The main features of our work are: (1) We focus on the class of *flexible* patterns. (2) We employ occurrence-based equivalence relations. (3) We propose two techniques of coarsening existing equivalence relations. (4) We develop a general algorithm GenCloFlex to handle several equivalence relations, and an improved algorithm GenCloFlex+. The algorithms have polynomial delay time and space guarantees.

(1) Mining flexible patterns: A *flexible* pattern is of the form $\star w_1 \star \cdots \star w_k \star$ where $w_1, \ldots, w_k$ $(k \geq 1)$ are constant strings and $\star$ is a gap symbol that can match any string of any length. Most studies to date target the class of *subsequence* patterns [1,3,5–16,20,21]. A subsequence pattern is a special case of flexible patterns, having the form $\star a_1 \star \cdots \star a_k \star$ $(k \geq 1)$ where each a_i $(1 \leq i \leq k)$ must be a single character. Thus, flexible patterns are more descriptive and enable us to capture some features that may not be discovered by subsequence patterns. For example, suppose we obtained $\star \mathtt{l} \star \mathtt{o} \star \mathtt{v} \star \mathtt{e} \star$ as an output of frequent subsequence pattern mining. Since the lengths of gaps between each character is not considered, the pattern does not distinguish its occurrences in texts “love” and “$\underline{\mathtt{lo}}\mathtt{w}$ $\underline{\mathtt{ve}}\mathtt{locity}$”, and we cannot know from the pattern alone, whether the phrase “love” is actually frequent or not. An output of frequent flexible pattern mining may give us the phrase “$\star\mathtt{love}\star$” or “$\star\mathtt{lo}\star\mathtt{ve}\star$” in which case this information would be apparent. Thus mining flexible patterns would be appreciated especially for languages which do not have an explicit delimiter between words such as Japanese and Chinese. Also, the information of consecutive characters in a pattern connected without gaps is very important for bio-sequences [18]. To the best of our knowledge, mining of closed flexible patterns with this definition has only been considered in [2]. A different version of closed flexible patterns is proposed in [19], but the definitions are significantly different and incompatible with ours.

(2) Occurrence-based equivalence relations: The definition of *closed* patterns depends on which equivalence relation to use, that is, which patterns we regard as the *same*. An equivalence relation is *finer* if less patterns are considered to be the same: i.e. more attention is paid to differences. An equivalence relation is *coarser* if more patterns are considered to be the same: i.e. less attention is paid to differences.

Most of the existing research on closed pattern mining traditionally use the equivalence relation on Π which is based on the *document occurrence*. Namely, two patterns are equivalent if the sets of strings in sequential database S containing occurrences of the patterns are identical. If a string T in database S contains an occurrence of P, we regard it as just one occurrence even if it contains two or more occurrences of P. For example, consider the occurrences of patterns $P_1 = \star\mathsf{a}\star\mathsf{b}\star\mathsf{cd}\star$ and $P_2 = \star\mathsf{a}\star\mathsf{cd}\star$ in string $T = \ldots\ldots\mathsf{a}\ldots\mathsf{b}..\mathsf{cd}\ldots\mathsf{a}\ldots\mathsf{cd}\ldots.$, where “.” denotes any symbol other than $\mathsf{a}, \mathsf{b}, \mathsf{c}$. We note that P_1 is a super-pattern of P_2 and therefore every occurrence of P_1 implies an occurrence of P_2, independently of strings in sequential database S. Suppose that every other string in S containing P_2 has an occurrence of P_1. Then the document-occurrence-based equivalence relation regards P_1 and P_2 equivalent. In this case, however, note that the rightmost occurrence of P_2 is not accompanied by an occurrence of P_1. In other words, if we consider the minimal occurrence intervals of the respective patterns, P_1 occurs twice while P_2 occurs only once within T. In some applications, we would like to distinguish between patterns which have different occurrences in such a way. In this paper we pay attention to respective occurrences of patterns, and use *occurrence-based* equivalence relations.

Arimura and Uno [2] defined their equivalence relation $\overset{\mathsf{B}}{\equiv}_S$ on pattern set Π based on the equality on the sets of beginning positions of pattern occurrences. For example, consider the occurrences of $P = \star\mathtt{a}\star\mathtt{b}\star$ in $T_1 =$ ...a.b...a...b..a.... The occurrence positions of P are the two occurrence positions of "a", excluding the rightmost one. However, we have four occurrence positions of P in $T_2 =$ ...a...a...a...a...b... while we have only one occurrence position of P in $T_3 =$ ...a...b...b...b...b.... Such a non-symmetric feature is usually not desirable.

Mannila *et al.* [9] defined their equivalence relation $\overset{\mathsf{M}}{\equiv}_S$ on the subsequence pattern set based on the minimal intervals within which patterns occur. In the previous example, the pattern P has only one occurrence in respective strings. In this paper we consider the closed pattern discovery problem mainly under $\overset{\mathsf{M}}{\equiv}_S$ extended to the flexible pattern set and its coarsened variants $\overset{\mathsf{MX}}{\equiv}_S$ and $\overset{\mathsf{MXG}}{\equiv}_S$.

It should be emphasized that occurrence-based equivalence relations such as $\overset{\mathsf{B}}{\equiv}_S$ and $\overset{\mathsf{M}}{\equiv}_S$ are finer than the document-occurrence-based equivalence relation. This means that using such equivalence relations may increase the number of mined closed patterns. Thus it is important to coarsen the equivalence relations.

On the other hand, the goodness of an equivalence relation may vary depending on the nature of the data, the application domain and the goal of pattern mining. For this reason, it is desirable to develop a general, efficient closed pattern mining algorithm for various equivalence relations.

(3) Definition of support: Note that the choice of equivalence relation does not imply a particular definition for the frequency, or *support*, of a pattern. There are several definitions of support of a pattern P in a sequential database S. One definition is the so-called *document frequency*, which is the number of strings in S that contain at least one occurrence of P. A lot of work on closed pattern mining employ this definition. Another definition is the sum of the numbers of minimal intervals in respective strings in S that contain at least one occurrence of P. This definition has been used, for example, in [9] and [22]. Yet another definition can be found in [9], which is the number of windows of a given width in respective strings in S that contain at least one occurrence of P.

Throughout this paper we assume the document frequency. However, our algorithms can be easily modified to cope with the other definitions above when the underlying equivalence relation is in the M family ($\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{MX}}{\equiv}_S$ and $\overset{\mathsf{MXG}}{\equiv}_S$).

(4) Polynomial delay time and space: Even in closed frequent pattern mining, the size of the output can be exponentially large, and we cannot hope for an algorithm running in polynomial time with respect to the input size. On the other hand, an enumeration algorithm with *polynomial delay time*, is an algorithm in which the time *between each consecutive output* is bounded by a polynomial with respect to the size of the input. Such characteristics can be very useful and important for mining algorithms, since it guarantees that the algorithm runs in polynomial time with respect to the size of the output. This means that the time complexity of the algorithm is small when the output size, i.e., the number of closed frequent patterns is small. Even when the output size is large, we can still expect that the next output can be received in a reasonable amount of time. Without this guarantee, we may find out – *after* waiting for a very long time – that there are no more frequent patterns to be discovered.

Space complexity of the mining algorithm is also clearly an important issue. Arimura and Uno [2] addressed the closed pattern discovery problem for the class of flexible patterns and presented the first algorithm MaxFlex with polynomial time delay and polynomial space. Our algorithms also achieve polynomial time delay and polynomial space.

For closed pattern mining under document-based equivalence relations, algorithms such as BIDE [16,17], proposed for subsequence patterns, seem to achieve polynomial space complexity. However, as far as we know, no time delay guarantees have been shown, which may be a consequence of the document-based equivalence relation.

Contributions of This Paper: We reiterate the main contributions of this paper: For the frequent closed flexible pattern enumeration problem, we focus on the equivalence relation $\overset{\mathsf{M}}{\equiv}_S$ of [9], and extended it to the class of flexible patterns. We also propose two new equivalence relations by coarsening it. We show GenCloFlex, an algorithm which generalizes the algorithm MaxFlex [2], so that it can cope with existing, as well as new equivalence relations, and investigate its computational complexities for respective equivalence relations. Then we present an improved algorithm GenCloFlex+, based on new pruning techniques which improve the delay time per output for some of the equivalence relations. By computational experiments on synthetic data, we prove that the proposed equivalence relations drastically remove redundancies in the mined patterns.

2 Preliminaries

Let Σ be a non-empty, finite set of symbols. A *string* over Σ is a finite sequence of symbols from Σ. Let Σ^* denote the set of strings over Σ. Strings x, y and z are said to be a *prefix*, *substring* and *suffix* of string $w = xyz$. The *length* of a string w is the number of symbols in w and denoted by $|w|$. The string of length 0 is called the *empty string* and denoted by ε. Let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. The i-th symbol of a string w is denoted by $w[i]$ for $1 \leq i \leq |w|$. The substring of a string w that begins at position i and ends at position j is denoted by $w[i..j]$ for $1 \leq i \leq j \leq |w|$. That is, $w[i..j] = w[i] \cdots w[j]$. For convenience, let $w[i..j] = \varepsilon$ for $j < i$. The *reversal* w^{rev} of a string $w = w[1..n]$ is defined to be $w[n] \cdots w[1]$. For a finite set S of strings, let $\|S\|$ denote the total length of strings in S and let $S^{rev} = \{w^{rev} \mid w \in S\}$.

An *interval* is an ordered pair $[i, j]$ of integers with $i \leq j$ which represents the set of integers k with $i \leq k \leq j$. Let I be a set of intervals. Let $\mathsf{Beg}(I) = \{i \mid$

$[i, j] \in I\}$ and $\mathsf{End}(I) = \{j \mid [i, j] \in I\}$, and let $\mathsf{Min}(I)$ denote the set of intervals in I which are minimal w.r.t. $\subseteq$. For any set I of intervals and for any integers h, k, let $I \oplus \langle h, k\rangle = \big\{[i + h, j + k] \bigm| [i, j] \in I\big\}$. Also, for any set J of integers and for any integer k, let $J \oplus k = \big\{j + k \bigm| j \in J\big\}$.

2.1 Equivalence Relation

Let A be a set. A *binary relation* on A is a subset of $A \times A$. For binary relations R_1, R_2 on A, let $R_1R_2 = \{\langle a, c\rangle \mid \langle a, b\rangle \in R_1 \wedge \langle b, c\rangle \in R_2\}$. Let $I_A = \{\langle a, a\rangle \mid a \in A\}$. For a binary relation R on A, let $R^0 = I_A$ and $R^n = RR^{n-1}$ $(n > 0)$, and let $R^{-1} = \{\langle b, a\rangle \mid \langle a, b\rangle \in R\}$, $R^+ = \bigcup_{n=1}^{\infty} R^n$ and $R^* = \bigcup_{n=0}^{\infty} R^n$.

A binary relation R on A is said to be *reflexive* if $I_A \subseteq R$; *symmetric* if $R^{-1} \subseteq R$; and *transitive* if $R^+ \subseteq R$. An *equivalence relation* on A is a binary relation on A which is reflexive, symmetric and transitive. For a binary relation R, we often write aRb when $\langle a, b\rangle \in R$.

Let $\equiv$ be an equivalence relation on A. The *equivalence class* of an element x of A under $\equiv$ is $\{y \in A \mid x \equiv y\}$ and denoted by $[x]_\equiv$. An equivalence relation $\equiv$ on A is said to be *finer* than another equivalence relation $\equiv'$ on A if $\equiv \subseteq \equiv'$. For any set $\mathcal{R} = \{\equiv_i \mid i \in \Lambda\}$ of equivalence relations on A, let $\wedge\mathcal{R} = \bigcap_{i\in\Lambda} \equiv_i$ and $\vee\mathcal{R} = \left(\bigcup_{i\in\Lambda} \equiv_i\right)^+$. The *equivalence closure* of a binary relation R on A, denoted by $\mathbf{EC}(R)$, is the smallest superset of R that is an equivalence relation on A. For any binary relation R on A, it is known that $\mathbf{EC}(R) = (R \cup R^{-1})^*$.

2.2 Pattern and Embedding

Let $\star$ be a special symbol not in Σ, called the *gap*. A *pattern* is of the form $\star w_1 \star \ldots \star w_k \star$ where $k \geq 1$ and $w_1, \ldots, w_k \in \Sigma^+$. Let Π be the set of patterns, and let Π_0 be the set of strings over $\Sigma \cup \{\star\}$ where the $\star$'s do not occur consecutively. We note that $\Pi \subset \Pi_0$. The *size* of a pattern P, denoted by *size*(P), is the number of symbols in P other than $\star$. The *reversal* of a pattern P, denoted by P^{rev}, is defined in the same way as in the string case. The *degree* of a pattern P is the number of occurrences of $\star$ in P and denoted by *deg*(P).

For example, let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. $P = \star\mathtt{ab}\star\mathtt{a}\star\mathtt{cb}\star$ is a flexible pattern. *size*$(P) = 5$, *deg*$(P) = 4$ and $P^{rev} = \star\mathtt{bc} \star \mathtt{a} \star \mathtt{ba}\star$.

A *substitution* of degree d is a d-tuple $\langle \pi_1, \ldots, \pi_d\rangle$ such that $\pi_1, \ldots, \pi_d \in \Pi_0$. A substitution $\langle \pi_1, \ldots, \pi_d\rangle$ is said to be *ground* if $\pi_1, \ldots, \pi_d \in \Sigma^*$. For any pattern $P \in \Pi$ and a substitution $\theta = \langle \pi_1, \ldots, \pi_d\rangle$ with $d = deg(P)$, *the application of θ to P*, denoted by $P\theta$, is the pattern obtained by replacing the i-th occurrence of $\star$ in P with π_i for every $i = 1, \ldots, d$. For any $i = 1, \ldots, d$, let Θ_d^i be the set of substitutions $\theta = \langle \pi_1, \ldots, \pi_d\rangle$ such that (1) $\pi_i \neq \star$ and (2) $\pi_j = \star$ for every j with $1 \leq j \leq d$ and $j \neq i$. Let $\Theta_d = \bigcup_{i=1}^{d} \Theta_d^i$. A substitution $\langle \star, \ldots, \star, \pi, \star, \ldots, \star\rangle \in \Theta_d^i$ is said to be *primitive* if $\pi \in \{\varepsilon\} \cup \{\star a \star \mid a \in \Sigma\}$. A primitive substitution $\langle \star, \ldots, \star, \pi, \star, \ldots, \star\rangle$ is said to be *erasing* if $\pi = \varepsilon$, and *non-erasing* if $\pi = \star a \star$ for some $a \in \Sigma$. For any $P, Q \in \Pi$, Q is said to be a *right-extension* of P if there exists a substitution $\theta \in \Theta_d^d$ such that $Q = P\theta$, where $d = deg(P)$.

An *embedding* of $P \in \Pi_0$ into $Q \in \Pi_0$ is a substitution θ such that $P\theta = Q$.

Definition 1. $P \lhd Q \overset{\text{def}}{\iff}$ *there is an embedding of* P *into* Q *that is primitive.*

Let $\lhd^*$ be a partial-order on Π_0 s.t. $P \lhd^* Q \iff$ there is an embedding of P into Q. For any $P \in \Pi$ and $T \in \Sigma^+$, P is said to *occur* in T if $P \lhd^+ T$.

For example, $\star\mathtt{ab}\star\mathtt{a}\star\mathtt{c}\star \lhd \star\mathtt{ab}\star\mathtt{d}\star\mathtt{a}\star\mathtt{c}\star$ due to a non-erasing primitive $\langle\star, \star\mathtt{d}\star, \star, \star\rangle \in \Theta_4$, $\star\mathtt{ab}\star\mathtt{a}\star\mathtt{c}\star \lhd \star\mathtt{ab}\star\mathtt{ac}\star$ due to an erasing primitive $\langle\star, \star, \varepsilon, \star\rangle \in \Theta_4$, and $\star\mathtt{ab}\star\mathtt{a}\star\mathtt{c}\star \lhd^* \star\mathtt{ab}\star\mathtt{d}\star\mathtt{ac}\star$ due to a substitution $\langle\star, \star\mathtt{d}\star, \varepsilon, \star\rangle$.

For an equivalence relation $\equiv$ on Π, a pattern $P \in \Pi$ is said to be *closed* under $\equiv$ if it is maximal in $[P]_\equiv$ w.r.t. $\lhd^*$.

2.3 Existing Equivalence Relations on Π

Let S be a finite subset of Σ^+. Intuitively, equivalence relations on Π are designed so that P and Q are equivalent if: *Every time* P *occurs in* S, Q *also occurs at the same location.* Difference between equivalence relations comes from the difference in definitions of *same location* here. Below, we describe several existing equivalence relations on Π.

An *occurrence interval* of a pattern $P \in \Pi$ in $T \in \Sigma^+$ is an interval $[|w_1| + 1, |T| - |w_d|]$ such that there is a ground embedding $\theta = \langle w_1, \ldots, w_d\rangle$ of P into T. Let $Int_T(P)$ be the set of all occurrence intervals of P in T. We give the definitions of four existing equivalence relations. Let S be a finite subset of Σ^+.

Definition 2 ($\overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{B}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S$)**.** *For any patterns* $P, Q \in \Pi$, *let*

$$\begin{aligned}
P \overset{\mathsf{I}}{\equiv}_S Q &\overset{\text{def}}{\iff} \forall T \in S,\ Int_T(P) = Int_T(Q),\\
P \overset{\mathsf{M}}{\equiv}_S Q &\overset{\text{def}}{\iff} \forall T \in S,\ \mathsf{Min}(Int_T(P)) = \mathsf{Min}(Int_T(Q)),\\
P \overset{\mathsf{B}}{\equiv}_S Q &\overset{\text{def}}{\iff} \forall T \in S,\ \mathsf{Beg}(Int_T(P)) = \mathsf{Beg}(Int_T(Q)),\\
P \overset{\mathsf{E}}{\equiv}_S Q &\overset{\text{def}}{\iff} \forall T \in S,\ \mathsf{End}(Int_T(P)) = \mathsf{End}(Int_T(Q)).
\end{aligned}$$

The algorithm MaxFlex [2] enumerates all closed flexible patterns in polynomial space and linear-time delay under $\overset{\mathsf{B}}{\equiv}_S$. $\overset{\mathsf{B}}{\equiv}_S$, $\overset{\mathsf{E}}{\equiv}_S$ and $\overset{\mathsf{B}}{\equiv}_S \vee \overset{\mathsf{E}}{\equiv}_S$ are natural extensions of the well-known equivalence relations introduced by Blumer et al. [4] for the class of substring patterns, which are recognized as the basis of index structures for text data, e.g., the suffix trees ($\overset{\mathsf{B}}{\equiv}_S$), the DAWGs ($\overset{\mathsf{E}}{\equiv}_S$), and the compact DAWGs ($\overset{\mathsf{B}}{\equiv}_S \vee \overset{\mathsf{E}}{\equiv}_S$). $\overset{\mathsf{M}}{\equiv}_S$ is an extension of the equivalence relation introduced by Mannila et al. [9] for the class of subsequence patterns.

Equivalence Relation Function: We note that the equivalence relations above vary depending on S. An *equivalence relation (ER) function* is a function that maps finite subsets S of Σ^+ to equivalence relations $\equiv_S$ on Π. The *reversal* of an ER function Φ is defined by: $\langle P, Q\rangle \in \Phi^{rev}(S) \iff \langle P^{rev}, Q^{rev}\rangle \in \Phi(S^{rev})$. We say that an ER function Φ is *symmetric* if $\Phi^{rev}(S) = \Phi(S)$ for every S. The ER functions for $\overset{\mathsf{I}}{\equiv}_S$, $\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{B}}{\equiv}_S \vee \overset{\mathsf{E}}{\equiv}_S$ and $\overset{\mathsf{B}}{\equiv}_S \wedge \overset{\mathsf{E}}{\equiv}_S$ are symmetric, while the reversal of the ER function for $\overset{\mathsf{B}}{\equiv}_S$ is the ER function for $\overset{\mathsf{E}}{\equiv}_S$, and vice versa.

Monotonicity of Equivalence Relations on Π**:** An equivalence relation $\equiv$ on Π is said to be *monotone* if $\lhd^+ \cap \equiv\ =\ (\lhd \cap \equiv)^+$. If $\equiv$ is monotone, then

$P_1 \lhd^+ P_2$ and $P_1 \equiv P_2$ implies that $\forall Q \in \Pi$, $(P_1 \lhd^* Q \lhd^* P_2 \implies P_1 \equiv Q \equiv P_2)$. Monotonicity of equivalence relations is a very helpful property for closedness check of a pattern, i.e., for any monotone equivalence relation $\equiv$, a pattern P is closed iff there is no primitive substitution θ such that $P\theta \equiv P$. We will discuss general and efficient algorithms based on monotonicity in Section 4. We remark that $\overset{\mathsf{I}}{\equiv}_S$, $\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{B}}{\equiv}_S$ and $\overset{\mathsf{E}}{\equiv}_S$ are monotone.

3 Coarsening Existing Equivalence Relations

Since we prefer symmetric equivalence relations, we focus on $\overset{\mathsf{M}}{\equiv}_S$ that is symmetric. It is, however, still too fine and we want a coarser one. In this section, we define two new equivalence relations $\overset{\mathsf{MX}}{\equiv}_S$ and $\overset{\mathsf{MXG}}{\equiv}_S$ by coarsening $\overset{\mathsf{M}}{\equiv}_S$. With one of the techniques, we also coarsen the other equivalence relations $\overset{\mathsf{I}}{\equiv}_S$, $\overset{\mathsf{B}}{\equiv}_S$ and $\overset{\mathsf{E}}{\equiv}_S$. and introduce $\overset{\mathsf{IX}}{\equiv}_S$, $\overset{\mathsf{BX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$.

For any pattern $P = \star w_1 \star \cdots \star w_k \star \in \Pi$, let $\overline{P} = w_1 \star \cdots \star w_k$, and thus $P = \star\overline{P}\star$. One technique of coarsening $\overset{\mathsf{M}}{\equiv}_S$ is to extend as long as possible the constant strings at pattern ends to the *outward* without decreasing occurrences.

Here we remark that the next proposition does hold.

Proposition 1. *For any $P \in \Pi$ and any substitution $\theta \in (\Theta_d^1 \cup \Theta_d^d)$ with $d = deg(P)$, $|\mathsf{Min}(Int_T(P))| \geq |\mathsf{Min}(Int_T(P\theta))|$ for every $T \in S$.*

Definition 3. $P \lhd_S^{\mathsf{MX}} Q \overset{\text{def}}{\iff}$ *there exists $a \in \Sigma$ such that*

- *$Q = \star\overline{P}a\star$ and $\mathsf{Min}(Int_T(Q)) = \mathsf{Min}(Int_T(P)) \oplus \langle 0, 1\rangle$ for every $T \in S$; or*
- *$Q = \star a\overline{P}\star$ and $\mathsf{Min}(Int_T(Q)) = \mathsf{Min}(Int_T(P)) \oplus \langle -1, 0\rangle$ for every $T \in S$.*

Definition 4 ($\overset{\mathsf{MX}}{\equiv}_S$). *Let $\overset{\mathsf{MX}}{\equiv}_S = \mathbf{EC}(\overset{\mathsf{M}}{\equiv}_S \cup \lhd_S^{\mathsf{MX}})$.*

Proposition 2. *For any $P \in \Pi$, there uniquely exists a pair of strings $u, v \in \Sigma^*$ such that $Q = \star u\overline{P}v\star$ is closed under $\overset{\mathsf{MX}}{\equiv}_S$ and $P \overset{\mathsf{MX}}{\equiv}_S Q$.*

Take an example S which consists of a single string $T_1 =$ `acbmdcacbndcaca`, and a pattern $P = \star\mathtt{b}\star\mathtt{d}\star$. There are two minimal occurrences of P in T_1, and we see they are preceded by "`ac`" and followed by "`cac`" without gap. Thus for any combinations of a suffix u of "`ac`" and a prefix v of "`cac`", $u\overline{P}v$ ($\star\mathtt{cb}\star\mathtt{dcac}\star$ for example) is equivalent to P under $\overset{\mathsf{MX}}{\equiv}_S$.

Another technique of coarsening $\overset{\mathsf{M}}{\equiv}_S$ is to add pattern fragments including gaps to the pattern ends without decreasing occurrences.

Definition 5. $P \lhd_S^{\mathsf{MXG}} Q \overset{\text{def}}{\iff} Q \in \{\star a \star \overline{P}\star, \star\overline{P} \star a\star\}$ *with some $a \in \Sigma$ and $|\mathsf{Min}(Int_T(P))| = |\mathsf{Min}(Int_T(Q))|$ for every $T \in S$.*

Definition 6 ($\overset{\mathsf{MXG}}{\equiv}_S$)**.** *Let* $\overset{\mathsf{MXG}}{\equiv}_S = \mathbf{EC}(\overset{\mathsf{M}}{\equiv}_S \cup \lhd_S^{\mathsf{MXG}})$.

Take an example S which consists of a single string $T_1 = \mathtt{acbmdcagcbnddcaca}$, and a pattern $P = \star\mathtt{b}\star\mathtt{d}\star$. The number of minimal occurrences of P in T_1 is 2. There are several patterns that are equivalent to P under $\overset{\mathsf{MXG}}{\equiv}_S$, but not under $\overset{\mathsf{M}}{\equiv}_S$ and $\overset{\mathsf{MX}}{\equiv}_S$, such as $\star\overline{P}\star\mathtt{c}\star$, $\star\mathtt{a}\star\mathtt{c}\overline{P}\star\mathtt{ca}\star\mathtt{c}\star\mathtt{a}\star$ and $\star\mathtt{a}\star\mathtt{c}\overline{P}\star\mathtt{d}\star$. Note that $Q = \star\overline{P}\star\mathtt{d}\star\mathtt{c}\star$ is not equivalent to P under $\overset{\mathsf{MXG}}{\equiv}_S$ because the number of minimal occurrences of Q is 1.

We can prove that $\overset{\mathsf{MX}}{\equiv}_S$ and $\overset{\mathsf{MXG}}{\equiv}_S$ are monotone from Proposition 1. Also, the ER functions for $\overset{\mathsf{MX}}{\equiv}_S$ and $\overset{\mathsf{MXG}}{\equiv}_S$ are symmetric. It follows from Definitions 4 and 6 that the next inclusion relation holds.

Theorem 1. $\overset{\mathsf{M}}{\equiv}_S \subseteq \overset{\mathsf{MX}}{\equiv}_S \subseteq \overset{\mathsf{MXG}}{\equiv}_S$.

The technique used in defining $\overset{\mathsf{MX}}{\equiv}_S$ extends $\overset{\mathsf{I}}{\equiv}_S$, $\overset{\mathsf{B}}{\equiv}_S$ and $\overset{\mathsf{E}}{\equiv}_S$ as below.

Definition 7. $P \lhd_S^{\mathsf{IX}} Q \overset{\text{def}}{\Longleftrightarrow}$ *there exists* $a \in \Sigma$ *such that*

- $Q = \star\overline{P}a\star$ *and* $\mathit{Int}_T(Q) = \mathit{Int}_T(P) \oplus \langle 0, 1\rangle$ *for every* $T \in S$*; or*
- $Q = \star a\overline{P}\star$ *and* $\mathit{Int}_T(Q) = \mathit{Int}_T(P) \oplus \langle -1, 0\rangle$ *for every* $T \in S$*.*

Definition 8. $P \lhd_S^{\mathsf{BX}} Q \overset{\text{def}}{\Longleftrightarrow}$ *there exists* $a \in \Sigma$ *such that* $Q = \star a\overline{P}\star$ *and* $\mathsf{Beg}(\mathit{Int}_T(Q)) = \mathsf{Beg}(\mathit{Int}_T(P)) \oplus (-1)$ *for every* $T \in S$*.*

Definition 9. $P \lhd_S^{\mathsf{EX}} Q \overset{\text{def}}{\Longleftrightarrow}$ *there exists* $a \in \Sigma$ *such that* $Q = \star\overline{P}a\star$ *and* $\mathsf{End}(\mathit{Int}_T(Q)) = \mathsf{End}(\mathit{Int}_T(P)) \oplus 1$ *for every* $T \in S$*.*

Definition 10 ($\overset{\mathsf{IX}}{\equiv}_S$, $\overset{\mathsf{BX}}{\equiv}_S$, $\overset{\mathsf{EX}}{\equiv}_S$)**.** *Let* $\overset{\mathsf{IX}}{\equiv}_S = \mathbf{EC}(\overset{\mathsf{I}}{\equiv}_S \cup \lhd_S^{\mathsf{IX}})$, $\overset{\mathsf{BX}}{\equiv}_S = \mathbf{EC}(\overset{\mathsf{B}}{\equiv}_S \cup \lhd_S^{\mathsf{BX}})$ *and* $\overset{\mathsf{EX}}{\equiv}_S = \mathbf{EC}(\overset{\mathsf{E}}{\equiv}_S \cup \lhd_S^{\mathsf{EX}})$.

We note that $\overset{\mathsf{IX}}{\equiv}_S$, $\overset{\mathsf{BX}}{\equiv}_S$, $\overset{\mathsf{EX}}{\equiv}_S$ are not monotone.

4 Algorithms for Enumerating Frequent Closed Patterns

Let $\mathit{Freq}_S(P)$ denote the number of strings in S in which P occurs.

*Problem 1 (*FreqCloPatEnum *w.r.t.* $\equiv$*).* Given a finite subset S of Σ^+ and a non-negative integer σ, enumerate all the patterns P closed under $\equiv$ without duplicates such that $\mathit{Freq}_S(P) \geq \sigma$.

Theorem 2 (MaxFlex [2]). *For a finite alphabet* Σ*, there exists an algorithm that solves* FreqCloPatEnum *w.r.t.* $\overset{\mathsf{B}}{\equiv}_S$ *in* $O(|\Sigma|\|S\|)$ *time delay and* $O(\|S\|d)$ *space, where* d *is the maximum number of gaps in the output patterns.*

In this section, we consider methods for efficiently solving the problem for the M family ($\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{MX}}{\equiv}_S$, $\overset{\mathsf{MXG}}{\equiv}_S$), the I family ($\overset{\mathsf{I}}{\equiv}_S$, $\overset{\mathsf{IX}}{\equiv}_S$), and the E family ($\overset{\mathsf{E}}{\equiv}_S$, $\overset{\mathsf{EX}}{\equiv}_S$). In the sequel, we exclude the descriptions for the B family ($\overset{\mathsf{B}}{\equiv}_S$, $\overset{\mathsf{BX}}{\equiv}_S$), since they are simply the reversal of the E family.

4.1 Outline of GenCloFlex

A pattern $P \in \Pi$ is said to be *i-th-gap-closed* under $\equiv$ if $P \not\equiv P\theta$ for every $\theta \in \Theta_d^i$, where $1 \leq i \leq d$ and $d = deg(P)$. For convenience, we say that P is *leftmost-gap-closed* for $i = 1$ and *rightmost-gap-closed* for $i = deg(P)$. A pattern $P \in \Pi$ is said to be *inner-gap-closed* if it is i-th-gap-closed under $\equiv$ for every i with $1 < i < deg(P)$.

An equivalence relation $\equiv$ on Π is said to be *rightmost-gap-independent* if any $P \in \Pi$ satisfies the following condition: For every i with $1 \leq i < deg(P)$, if P is not i-th-gap-closed under $\equiv$, then every right-extension P' of P is not i-th-gap-closed under $\equiv$. We remark that the M, I and E families are rightmost-gap-independent.

As a generalization of MaxFlex [2], we describe GenCloFlex, an algorithm for solving FreqCloPatEnum w.r.t. *any* equivalence relation that is rightmost-gap-independent. We define a rooted search-tree **ST** over $\Pi \cup \{\bot\}$ by:

- For any $a \in \Sigma$, the parent of $\star a \star$ is $\bot$.
- For any $a \in \Sigma$ and for any $w_1, \ldots, w_k \in \Sigma^+$, the parent of $\star w_1 \star \cdots \star w_k a \star$ and $\star w_1 \star \cdots \star w_k \star a \star$ is $\star w_1 \star \cdots \star w_k \star$.

Lemma 1. *For any $P, Q \in \Pi$, $P \lhd^* Q$ implies $Freq_S(P) \geq Freq_S(Q)$.*

Lemma 2 (general pruning rule). *Let $\equiv$ be any rightmost-gap-independent equivalence relation on Π. Under $\equiv$, if $P \in \Pi$ is not i-th-gap-closed for some i with $1 \leq i < deg(P)$, then no descendant of P in* **ST** *is closed.*

Algorithm 1 outlines a general algorithm for FreqCloPatEnum under *any* rightmost-gap-independent equivalence relation. The algorithm performs a depth-first-traversal of **ST**, with pruning based on Lemmas 1 and 2. We note that the algorithm does not build **ST** actually.

4.2 Closedness Tests

We now consider how to realize the inner-, the leftmost- and the rightmost-closedness tests. An equivalence relation $\equiv$ on Π is said to be *inner-gap-monotone* if for any $P \in \Pi$ and any $\theta \in \Theta_d - (\Theta_d^1 \cup \Theta_d^d)$ with $d = deg(P)$, $P \equiv P\theta$ implies that $\forall Q \in \Pi$, $(P \lhd^* Q \lhd^* P\theta \implies P \equiv Q \equiv P\theta)$. We remark that the M, I and E families are all inner-gap-monotone.

Lemma 3 (inner-gap-closedness test). *Let $\equiv$ be any inner-gap-monotone equivalence relation on Π. Let $P \in \Pi$. Then, for any i with $1 < i < deg(P)$, P is i-th-gap-closed under $\equiv$ if $P \not\equiv P\theta$ for every primitive substitution θ in Θ_d^i.*

For a monotone equivalence relation $\equiv$, the leftmost- (resp. rightmost-) gap-closedness of $P \in \Pi$ can also be tested by checking whether $P \not\equiv P\theta$ for every non-erasing primitive substitution θ in Θ_d^1 (resp. Θ_d^d). For $\overset{\mathsf{IX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$ that are not monotone, we have the following lemma:

Algorithm 1. General Algorithm GenCloFlex for FreqCloPatEnum

```
Input: a finite subset $S$ of $\Sigma^+$ and a non-negative integer $\sigma$.
Output: non-duplicate list of patterns $P$ with $Freq_S(P) \geq \sigma$ that are closed
        under a rightmost-gap-independent equivalence relation $\equiv$.
foreach $a \in \Sigma$ do Expand($\star a \star$);

procedure Expand($P$);
let $d := deg(P)$;
if $Freq_S(P) < \sigma$ then return;
if ($P$ is not leftmost-gap-closed) or ($P$ is not inner-gap-closed) then
    return; // Pruning
// Now $P$ is leftmost-gap-closed and inner-gap-closed
if $P$ is rightmost-gap-closed then // $P$ is closed
    report $P$;
foreach $a \in \Sigma$ do
    Expand($\star\overline{P} \star a\star$);
    Expand($\star\overline{P}a\star$);
```

Lemma 4 (leftmost-, rightmost-gap-closedness tests for $\overset{\mathsf{IX}}{\equiv}_S$, $\overset{\mathsf{EX}}{\equiv}_S$, $\overset{\mathsf{MX}}{\equiv}_S$). *Let* $\equiv \in \{\overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S\}$ *and* $P \in \Pi$. *P is leftmost-gap-closed under* $\equiv$ *if* $P \not\equiv \star a\overline{P}\star$ *for every* $a \in \Sigma$. *P is rightmost-gap-closed under* $\equiv$ *if* $P \not\equiv \star\overline{P}a\star$ *for every* $a \in \Sigma$.

Lemma 5. *The time complexities of the leftmost-, the rightmost- and the inner-gap-closedness tests for* $P \in \Pi$ *with* $d = deg(P)$ *are summarized in Table 1.*

Proof. The leftmost- and the rightmost-gap-closedness tests for $\overset{\mathsf{I}}{\equiv}_S$ and $\overset{\mathsf{M}}{\equiv}_S$ and the rightmost-gap-closedness test for $\overset{\mathsf{E}}{\equiv}_S$ are unnecessary by their definitions. The leftmost-gap-closedness test for $\overset{\mathsf{E}}{\equiv}_S$ takes $O(\|S\|)$ time as shown in [2]. By Lemma 4 the leftmost-gap-closedness test (resp. the rightmost-gap-closedness test) for $\overset{\mathsf{MX}}{\equiv}_S$ can be performed simply by checking whether all minimal occurrences of P are directly preceded by (resp. followed by) a same symbol. This takes $O(\|S\|)$ time. Similarly, the rightmost-gap-closedness test for $\overset{\mathsf{EX}}{\equiv}_S$ and the leftmost- and the rightmost-gap-closedness test for $\overset{\mathsf{IX}}{\equiv}_S$ take $O(\|S\|)$ time. Now we consider the rightmost-gap-closedness test for $\overset{\mathsf{MXG}}{\equiv}_S$. What we have to do is to check whether there exists a non-erasing primitive substitution θ in Θ_d^d which preserves $|\mathsf{Min}(Int_T(P))|$ in every $T \in S$. Let $e_1, \ldots, e_m$ be the increasing sequence of ending positions of $\mathsf{Min}(Int_T(P))$. Let $I_i = [e_1 + 1, e_{i+1}]$ for $i = 1, \ldots, m-1$ and let $I_m = [e_m + 1, |T|]$. We can build the list of symbols common to the substrings of T implied by $I_1, \ldots, I_m$ in $O(|T|)$ time. The test thus takes $O(\|S\|)$ time. The leftmost-gap-closedness test also takes $O(\|S\|)$ time.

We now suppose $d > 2$ for the inner-gap-closedness test. For the E family, it suffices to determine whether $P\theta$ occurs within the leftmost occurrence interval of P. This can be done in $O(\|S\|)$ time independently of d by using an auxiliary

Table 1. Time complexities of the leftmost-, the rightmost- and the inner-gap-closedness tests for respective equivalence relations

	leftmost-gap-closedness	rightmost-gap-closedness	inner-gap-closedness
$\overset{\mathsf{I}}{\equiv}_S$	(always true)	(always true)	$O(\|S\|d)$
$\overset{\mathsf{IX}}{\equiv}_S$	$O(\|S\|)$	$O(\|S\|)$	$O(\|S\|d)$
$\overset{\mathsf{M}}{\equiv}_S$	(always true)	(always true)	$O(\|S\|d)$
$\overset{\mathsf{MX}}{\equiv}_S$	$O(\|S\|)$	$O(\|S\|)$	$O(\|S\|d)$
$\overset{\mathsf{MXG}}{\equiv}_S$	$O(\|S\|)$	$O(\|S\|)$	$O(\|S\|d)$
$\overset{\mathsf{E}}{\equiv}_S$	$O(\|S\|)$	(always true)	$O(\|S\|)$
$\overset{\mathsf{EX}}{\equiv}_S$	$O(\|S\|)$	$O(\|S\|)$	$O(\|S\|)$

Table 2. Delay time per output for respective equivalence relations

	GenCloFlex	GenCloFlex+
$\overset{\mathsf{I}}{\equiv}_S$	$O(\|\Sigma\|\|S\|d)$	$O(\|\Sigma\|\|S\|d)$
$\overset{\mathsf{IX}}{\equiv}_S$	$O(\|\Sigma\|\|S\|^2d)$	$O(\|\Sigma\|\|S\|d)$
$\overset{\mathsf{M}}{\equiv}_S$	$O(\|\Sigma\|\|S\|d)$	$O(\|\Sigma\|\|S\|d)$
$\overset{\mathsf{MX}}{\equiv}_S$	$O(\|\Sigma\|\|S\|^2d)$	$O(\|\Sigma\|\|S\|d)$
$\overset{\mathsf{MXG}}{\equiv}_S$	$O(\|\Sigma\|\|S\|^2d)$	$O(\|\Sigma\|\|S\|^2d)$
$\overset{\mathsf{E}}{\equiv}_S$	$O(\|\Sigma\|\|S\|)$ [2]	$O(\|\Sigma\|\|S\|)$
$\overset{\mathsf{EX}}{\equiv}_S$	$O(\|\Sigma\|\|S\|^2)$	$O(\|\Sigma\|\|S\|)$

data structure of size $O(\|S\|d)$ as shown in [2]. For the M family, we have to check it over all minimal occurrence intervals of P, and the same technique cannot be applied. This takes $O(\|S\|d)$ time and space. For the I family, we basically check it over all occurrence intervals of P. For the erasing primitive substitution θ in Θ_d^2 (resp. Θ_d^{d-1}), it suffices to check whether $P\theta$ begins (resp. ends) at every beginning (resp. ending) positions of occurrence intervals of P. For the other erasing primitive substitutions or for the non-erasing primitive substitutions, it suffices to consider only the minimal occurrence intervals. □

4.3 Improved Algorithms for Respective Equivalence Relations

We introduce new efficient pruning techniques based on common extensions. Especially, the techniques improve the time complexity for $\overset{\mathsf{MX}}{\equiv}_S$, $\overset{\mathsf{IX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$, as shown in Table 2.

Let $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$. The *longest common extension* of $P \in \Pi$ under $\equiv$ is the longest string $v \in \Sigma^*$ such that for every $T \in S$,

- $\mathsf{Min}(Int_T(\star\overline{P}v\star)) = \mathsf{Min}(Int_T(P)) \oplus \langle 0, |v|\rangle$ when $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S\}$;
- $Int_T(\star\overline{P}v\star) = Int_T(P) \oplus \langle 0, |v|\rangle$ when $\equiv \in \{\overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$.

When $v \neq \varepsilon$, $c = v[1]$ is said to be the *common extension* of P under $\equiv$.

The following lemmas help us to skip unnecessary closedness tests.

Lemma 6 (skipping leftmost- and inner-gap-closedness tests). *Let* $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$*. If* P *is leftmost- and inner-gap-closed under* $\equiv$, $\star\overline{P}c\star$ *is also leftmost- and inner-gap-closed under* $\equiv$*.*

Lemma 7 (skipping rightmost-gap-closedness tests). *Let* $\equiv \in \{\overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$*. Then* P *is not rightmost-gap-closed under* $\equiv$*.*

For the M family, we can utilize the following lemma for pruning.

Lemma 8 (pruning for the M family). *Let* $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$*. Then among the descendants of* P *in* $\mathbf{\mathsf{ST}}$*, only descendants of* $\star\overline{P}c\star$ *can be closed under* $\equiv$*.*

Proof. Since $\mathsf{Min}(\mathit{Int}_T(\star\overline{P}c\star)) = \mathsf{Min}(\mathit{Int}_T(P)) \oplus \langle 0, 1\rangle$ for every $T \in S$, $\star\overline{P}\star c\star$ is not closed due to $\star\overline{P}\star c\star \equiv \star\overline{P}c\star$. Since $\langle \star\overline{P} \star a\star, \star\overline{P}c \star a\star\rangle \in \lhd^+ \cap \equiv$, $\star\overline{P} \star a\star$ is not closed for any $a \in \Sigma - \{c\}$. Let $P = \star w_1 \star \cdots \star w_k\star$, where $w_1, \ldots, w_k \in \Sigma^+$. Let $P' = \star w_1 \star \cdots \star w_{k-1} \star b \star w_k a\star$, where b is the first symbol of w_k. Since $\langle \star\overline{P}a\star, P'\rangle \in \lhd^+ \cap \equiv$, $\star\overline{P}a\star$ is not closed for any $a \in \Sigma - \{c\}$. Since $\equiv$ is monotone, the lemma holds. □

For the I and E families, we can utilize the following lemmas for pruning.

Lemma 9 (pruning for the I and E families). *Let* $\equiv \in \{\overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$*. Then among the descendants of* P *in* $\mathbf{\mathsf{ST}}$*, only descendants of* $\star\overline{P}c\star$ *and of* $\star\overline{P}\star c\star$ *can be closed under* $\equiv$*.*

Proof. $\star\overline{P}a\star$ does not occur in S for any $a \in \Sigma - \{c\}$. Since $\langle \star\overline{P}\star a\star, \star\overline{P}c\star a\star\rangle \in \lhd^+ \cap \equiv$, $\star\overline{P}\star a\star$ is not inner-gap-closed for any $a \in \Sigma - \{c\}$. Since $\equiv$ is inner-gap-monotone, the lemma holds. □

Lemma 10. *Let* $\equiv \in \{\overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$*.* $|\mathsf{End}(\mathit{Int}_T(P))| = |\{i \mid \min\{\mathsf{End}(\mathit{Int}_T(P))\} < i \leq |T|, T[i] = c\}|$ *for every* $T \in S \iff \star\overline{P}\star c\star \equiv \star\overline{P}c\star$*.*

Proof. Since c is the common extension of P under $\equiv$, $\mathit{Int}_T(\star\overline{P}c\star) = \mathit{Int}_T(P) \oplus \langle 0, 1\rangle$ for every $T \in S$. Adding to this, the left-hand condition implies that $\mathit{Int}_T(\star\overline{P}\star c\star) = \mathit{Int}_T(\star\overline{P}c\star)$ for every $T \in S$. Hence the $\Longrightarrow$ statement holds. The $\Longleftarrow$ statement follows from the fact that $|\mathsf{End}(\mathit{Int}_T(\star\overline{P}\star c\star))| > |\mathsf{End}(\mathit{Int}_T(\star\overline{P}c\star))|$ for some $T \in S$ if the left-hand condition does not hold. □

For any $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ and $P \in \Pi$, the common extension c of P under $\equiv$ is said to *make a branch* if $\star\overline{P}\star c\star \not\equiv \star\overline{P}c\star$. Clearly from Lemma 8, any common extension does not make a branch for the M family. For the I and E families, it follows from Lemma 10 that a common extension c makes a branch iff $|\mathsf{End}(\mathit{Int}_T(P))| < |\{i \mid \min\{\mathsf{End}(\mathit{Int}_T(P))\} < i \leq |T|, T[i] = c\}|$ for some $T \in S$.

The algorithm based on Lemmas 6, 7, 8, 9 and 10 can be summarized as Algorithm 2. We remark that the longest common extension v can be represented in constant space, by the pair of a pointer to some position in $T \in S$ where v occurs and length $|v|$.

4.4 Time Complexities

Theorem 3 (GenCloFlex,GenCloFlex+). *For a finite alphabet* Σ*, Algorithms 1 and 2 solve* FreqCloPatEnum *for respective equivalence relations with time delay shown in Table 2 and* $O(\|S\|d)$ *space, where* d *is the maximum number of gaps in the output patterns.*

Algorithm 2. Improved Algorithm GenCloFlex+ for FreqCloPatEnum

```
   Input: a finite subset $S$ of $\Sigma^+$ and a non-negative integer $\sigma$.
   Output: non-duplicate list of patterns $P$ with $Freq_S(P) \geq \sigma$ that are closed
           under $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$.
 1 foreach $a \in \Sigma$ do CheckExtension($\star a \star$);

   procedure ExpandWithCommonExtension($P$, $v$);
 1 if $P$ is rightmost-gap-closed then // When $|v| > 0$, use Lemma 7
 2     report $P$;
 3 if $|v| > 0$ then
 4     let $c := v[1]$;
 5     ExpandWithCommonExtension($\star\overline{P}c\star$, $v[2..|v|]$);
 6     if $c$ makes a branch then
 7         CheckExtension($\star\overline{P}\star c\star$);
 8 else
 9     foreach $a \in \Sigma$ do
10         CheckExtension($\star\overline{P}\star a\star$);
11         CheckExtension($\star\overline{P}a\star$);

   procedure CheckExtension($P$);
 1 if $Freq_S(P) < \sigma$ then return;
 2 if ($P$ is not leftmost-gap-closed) or ($P$ is not inner-gap-closed) then
 3     return; // Pruning
   // Now $P$ is leftmost-gap-closed and inner-gap-closed
 4 compute the longest common extension $v$ of $P$ under $\equiv$;
 5 ExpandWithCommonExtension($P$, $v$);
```

Proof. Let $CT_\equiv$ denote the cost of the closedness test under $\equiv$, i.e., $CT_\equiv \in O(\|S\|)$ if $\equiv$ is the E family, $CT_\equiv \in O(\|S\|d)$ if $\equiv$ is the M or I family.

For GenCloFlex: For $\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{I}}{\equiv}_S$ and $\overset{\mathsf{E}}{\equiv}_S$, the rightmost-gap-closedness test is unnecessary and therefore the condition of the **if**-statement at Line 6 is always satisfied and pattern P is always reported. As a result, the delay time per output is obtained by $O(|\Sigma| \times CT_\equiv)$. On the other hand, for $\overset{\mathsf{MX}}{\equiv}_S$, $\overset{\mathsf{MXG}}{\equiv}_S$, $\overset{\mathsf{IX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$, pattern P is reported only when the condition is satisfied. Nevertheless, we can find a frequent closed pattern after going down **ST** at most $\|S\|$ unreported patterns, and hence, the delay for output is $O(|\Sigma|\|S\| \times CT_\equiv)$.

For GenCloFlex+: At Line 4 of Procedure CheckExtension(P), we compute the longest common extension of P under $\equiv$. It is equivalent to compute the longest common prefix of $\bigcup_{T\in S}\{T[j+1..|T|] \mid j \in \mathsf{End}(\mathsf{Min}(Int_T(P)))\}$ (resp. $\bigcup_{T\in S}\{T[j+1..|T|] \mid j \in \mathsf{End}(Int_T(P))\}$) for the M family (resp. for the I and E families), and hence, is done in $O(\|S\|)$ time.

In the case of the I or E family, we also compute the positions where the common extension makes a branch as follows.

For every $T \in S$ in which P occurs, do the following:

1. Compute $F(a) = |\{i \mid \min\{\mathsf{End}(Int_T(P))\} < i \leq |T|, T[i] = a\}|$ for any symbol a used in v.
2. For each $j = 1, \ldots, |v|$ in increasing order, do the following:
   (a) If $|\mathsf{End}(Int_T(P))| < F(v[j])$ then $v[j]$ makes a branch.
   (b) Decrement $F(v[j])$ by 1.

Thus we can compute the branching positions in v in $O(\|S\|)$ time.

Here we estimate the delay time per output for $\equiv \in \{\overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$. Let us consider the call `CheckExtension`(P) such that P is not rightmost-gap-closed, i.e., there exists $c \in \Sigma$ with $P \equiv \star\overline{P}c\star$. From the definition of $\overset{\mathsf{MX}}{\equiv}_S$, $\overset{\mathsf{IX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$, $\star\overline{P}v\star$ is closed, where v is the longest common extension of P under $\equiv$. Since v can be computed in $O(\|S\|)$ time and the closedness test for P takes $O(CT_\equiv)$ time, we can output $\star\overline{P}v\star$ in $O(CT_\equiv)$ time just after `CheckExtension`(P) is executed. Hence the delay from a previous output to $\star\overline{P}v\star$ is $O(|\Sigma| \times CT_\equiv)$. □

The closedness checks for $\overset{\mathsf{MXG}}{\equiv}_S$ take a constant factor more time than those for $\overset{\mathsf{M}}{\equiv}_S$ and $\overset{\mathsf{MX}}{\equiv}_S$. However, it should be noted that the total theoretical asymptotic worst case time complexities for $\overset{\mathsf{MXG}}{\equiv}_S$ is equal to or smaller than those for $\overset{\mathsf{M}}{\equiv}_S$ and $\overset{\mathsf{MX}}{\equiv}_S$, due to the smaller search space for $\overset{\mathsf{MXG}}{\equiv}_S$ in **ST**. Hence, $\overset{\mathsf{MXG}}{\equiv}_S$ never falls far behind $\overset{\mathsf{M}}{\equiv}_S$ and $\overset{\mathsf{MX}}{\equiv}_S$, and can be much faster.

5 Computational Experiments

We implemented our algorithms for I, M and E families in the C language. Recall that $\overset{\mathsf{B}}{\equiv}_S$ is just the reversal of $\overset{\mathsf{E}}{\equiv}_S$, and thus, GenCloFlex $\overset{\mathsf{E}}{\equiv}_S$ can essentially be regarded as MaxFlex [2]. Considering the trade-off in implementation, we used naive matching to compute the longest common extensions, and did not implement the pruning technique based on Lemma 10. All the computational experiments were carried out on Apple Xserve with two Quad-Core Intel Xeon at 2.93GHz (8 CPU x 2 HT), with 24GB Memory 1066MHz DDR3.

We carried out experiments on synthetic data. To create data sets for examining flexible pattern mining algorithms, we modified IBM sequence generator [1], which is widely used in the subsequence pattern mining research area [3,5–8,13,20]. The original program generates random sequences of item sets and embeds copies of some item set sequence as a pattern which is randomly corrupted. Although, originally, each item set is sorted and represented as a sorted integer sequence, we use the unsorted sequence representation. Each such sequence in the pattern is considered as a segment of the flexible pattern. In this way, we are able to generate a data set of integer strings in which some flexible patterns are embedded, where each segment is damaged in the same manner as the original program (See [1] for more details).

There are several parameters: [D] number of generated strings in 1000s, [C] average length of strings, [N] alphabet size in 1000s, [P] number of patterns, [L] average number of segments of patterns and [S] average length of segments of

Table 3. Experiments on Synthetic Data Sets (threshold value is fixed to $\sigma = 10$)

	D5C40N1P500L4S2		D5C40N1P500L4S4		D5C40N1P500L4S6	
algorithm/equiv	patterns	seconds	patterns	seconds	patterns	seconds
GenCloFlex $\overset{\mathsf{I}}{\equiv}_S$	271,412	1155	216,513	1119	201,746	972
GenCloFlex+ $\overset{\mathsf{I}}{\equiv}_S$	271,412	1105 (96%)	216,513	967 (86%)	201,746	800 (82%)
GenCloFlex+ $\overset{\mathsf{IX}}{\equiv}_S$	255,910 (94%)	1104 (96%)	182,868 (84%)	970 (87%)	159,898 (79%)	777 (80%)
GenCloFlex $\overset{\mathsf{M}}{\equiv}_S$	294,183	1225	285,954	1434	305,893	1360
GenCloFlex+ $\overset{\mathsf{M}}{\equiv}_S$	294,183	1161 (95%)	285,954	1116 (78%)	305,893	948 (70%)
GenCloFlex+ $\overset{\mathsf{MX}}{\equiv}_S$	253,928 (86%)	1075 (88%)	180,576 (63%)	928 (65%)	162,658 (53%)	781 (57%)
GenCloFlex+ $\overset{\mathsf{MXG}}{\equiv}_S$	247,036 (84%)	1060 (87%)	165,175 (58%)	874 (61%)	143,930 (47%)	731 (54%)
GenCloFlex $\overset{\mathsf{E}}{\equiv}_S$	311,526	1380	328,368	1717	341,138	1545
GenCloFlex+ $\overset{\mathsf{E}}{\equiv}_S$	311,526	1266 (92%)	328,368	1368 (80%)	341,138	1100 (71%)
GenCloFlex+ $\overset{\mathsf{EX}}{\equiv}_S$	270,511 (87%)	1180 (86%)	209,971 (64%)	1146 (67%)	184,289 (54%)	904 (59%)

patterns. We omitted scalability tests since we clearly showed time complexities of our algorithms. Instead we are interested in how the parameter [S] affects efficiency of our algorithms, and therefore experimented on three data sets with parameters D5C40N1P500L4S2, D5C40N1P500L4S4 and D5C40N1P500L4S6.

In Table 3, we compare the number of output closed patterns and computational time, where xx% is the relative ratio compared to GenCloFlex of respective families. The result shows that redundant output patterns which can be removed by our coarsened equivalence relations increase as the average length of frequent segments embedded in a data set becomes longer. Thus our algorithms with coarsened equivalence relations would be effective especially for data which is expected to contain long frequent segments such as bio-sequences.

6 Conclusion

We addressed the closed pattern discovery problem for the class of flexible patterns. We focused on the minimal-occurrence-interval based equivalence relation $\overset{\mathsf{M}}{\equiv}_S$ on the set of patterns introduced by Mannila et al. [9], and proposed two new equivalence relations by coarsening it. We investigated the properties of equivalence relations on the patterns from viewpoints of closed pattern enumeration, and as a generalization of the algorithm proposed by Arimura and Uno [2], we showed a general algorithm for enumerating closed patterns for existing and newly proposed equivalence relations of various kinds. Then we accelerated the algorithm by a set of new pruning techniques. Computational experiments on synthetic data implied that the proposed equivalence relations successfully remove some redundancy in the output patterns compared to the existing equivalence relations. Finally, although the results are not shown due to space limitation, we applied our algorithm on real data: Waka poems – traditional Japanese poetry with over 1300-year history – and confirmed that our algorithms with coarsened equivalence relations output reasonable amounts of mined patterns and increase their readability.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE, pp. 3–14 (1995)
2. Arimura, H., Uno, T.: Mining Maximal Flexible Patterns in a Sequence. In: Satoh, K., Inokuchi, A., Nagao, K., Kawamura, T. (eds.) JSAI 2007. LNCS (LNAI), vol. 4914, pp. 307–317. Springer, Heidelberg (2008)
3. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: KDD, pp. 429–435 (2002)
4. Blumer, A., Blumer, J., Haussler, D., McConnell, R., Ehrenfeucht, A.: Complete inverted files for efficient text retrieval and analysis. J. ACM 34(3), 578–595 (1987)
5. Ding, B., Lo, D., Han, J., Khoo, S.-C.: Efficient mining of closed repetitive gapped subsequences from a sequence database. In: ICDE, pp. 1024–1035 (2009)
6. Lo, D., Cheng, H.: Lucia: Mining closed discriminative dyadic sequential patterns. In: EDBT, pp. 21–32 (2011)
7. Lo, D., Ding, B., Lucia, Han, J.: Bidirectional mining of non-redundant recurrent rules from a sequence database. In: ICDE, pp. 1043–1054 (2011)
8. Lo, D., Khoo, S.C., Li, J.: Mining and ranking generators of sequential patterns. In: SDM, pp. 553–564 (2008)
9. Mannila, H., Toivonen, H., Verkamo, I.A.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3), 259–289 (1997)
10. Parida, L., Rigoutsos, I., Floratos, A., Platt, D.E., Gao, Y.: Pattern discovery on character sets and real-valued data: linear bound on irredundant motifs and an efficient polynomial time algorithm. In: Proc. SODA, pp. 297–308 (2000)
11. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns by prefix-projected growth. In: ICDE, pp. 215–224 (2001)
12. Pisanti, N., Crochemore, M., Grossi, R., Sagot, M.-F.: A basis of tiling motifs for generating repeated patterns and its complexity for higher quorum. In: Rovan, B., Vojtáš, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 622–631. Springer, Heidelberg (2003)
13. Raïssi, C., Calders, T., Poncelet, P.: Mining Conjunctive Sequential Patterns. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, p. 19. Springer, Heidelberg (2008)
14. Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
15. Tatti, N., Cule, B.: Mining closed strict episodes. In: ICDM, pp. 501–510 (2010)
16. Wang, J., Han, J.: BIDE: Efficient mining of frequent closed sequences. In: ICDE, pp. 79–90 (2004)
17. Wang, J., Han, J., Li, C.: Frequent closed sequence mining without candidate maintenance. IEEE Transactions on Knowledge and Data Engineering 19(8), 1042–1056 (2007)
18. Wang, K., Xu, Y., Yu, J.X.: Scalable sequential pattern mining for biological sequences. In: CIKM, pp. 178–187 (2004)
19. Wu, H.-W., Lee, A.J.T.: Mining closed flexible patterns in time-series databases. Expert Systems with Applications 37(3), 2098 (2010)
20. Yan, X., Han, J., Afshar, R.: CloSpan: Mining closed sequential patterns in large databases. In: SDM (2003)
21. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning 42(1/2), 31–60 (2001)
22. Zhou, W., Liu, H., Cheng, H.: Mining Closed Episodes from Event Sequences Efficiently. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part I. LNCS, vol. 6118, pp. 310–318. Springer, Heidelberg (2010)

Size Matters: Finding the Most Informative Set of Window Lengths*

Jefrey Lijffijt[1], Panagiotis Papapetrou[1,2], and Kai Puolamäki[1]

[1] Department of Information and Computer Science, Aalto University, Finland
[2] Department of Computer Science and Information Systems, Birkbeck, University of London, UK

Abstract. Event sequences often contain continuous variability at different levels. In other words, their properties and characteristics change at different rates, concurrently. For example, the sales of a product may slowly become more frequent over a period of several weeks, but there may be interesting variation within a week at the same time. To provide an accurate and robust "view" of such multi-level structural behavior, one needs to determine the appropriate levels of granularity for analyzing the underlying sequence. We introduce the novel problem of finding the best set of window lengths for analyzing discrete event sequences. We define suitable criteria for choosing window lengths and propose an efficient method to solve the problem. We give examples of tasks that demonstrate the applicability of the problem and present extensive experiments on both synthetic data and real data from two domains: text and DNA. We find that the optimal sets of window lengths themselves can provide new insight into the data, e.g., the burstiness of events affects the optimal window lengths for measuring the event frequencies.

Keywords: event sequence, window length, clustering, exploratory data mining.

1 Introduction

Many sequences involve slowly changing properties, mixed with faster changing properties. For example, the sales of a product may slowly become more frequent over a period of several weeks, but there may be interesting variation throughout a week at the same time. To provide an accurate and robust "view" of such multi-level structural behavior, one needs to determine the appropriate levels of granularity for analyzing the underlying sequence.

Sliding windows are frequently employed in several sequence analysis tasks, such as mining frequent episodes [25], discovering poly-regions in biological sequences [29], finding biological or time-series motifs [6,7], analysis of electroencephalogram (EEG) sequences [30], or in linguistic analysis of documents [3]. A major problem is that such methods are often parametrized by a user-defined

* This work was supported by the Finnish Centre of Excellence for Algorithmic Data Analysis Research (ALGODAN).

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 451–466, 2012.

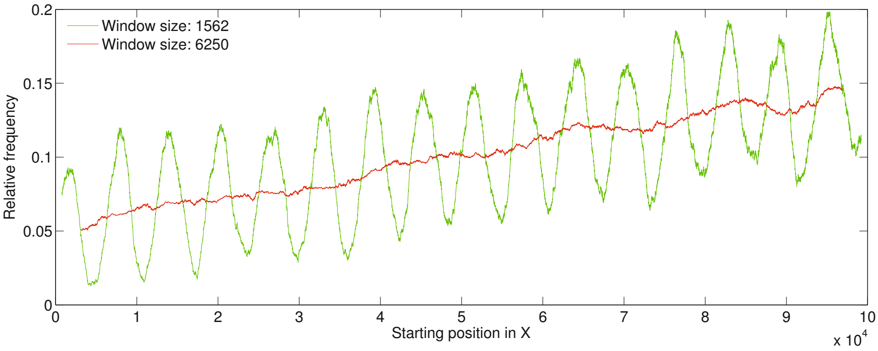


Fig. 1. Frequency change of an event in a sequence X of length 100,000. We use two sliding windows of lengths 1562 and 6250. The generative process for this sequence is described in Section 5.2. We observe that the two window lengths describe two different views of the data.

window length and it can be unclear how to choose the appropriate window length that guarantees optimal execution of the task at hand.

This problem can be avoided by either (i) defining an appropriate objective function and using an optimization algorithm to select the best window length, or (ii) using all possible window lengths at the same time. The first approach has, on one hand, the limitation that a single window length may leave out important information that would be discovered when using other window lengths. On the other hand, studying all window lengths does not have this deficiency, however, it may be challenging to analyze the large amount of information. The method proposed in this paper is to use a small set of window lengths that together provide as much information as possible about the underlying data. We demonstrate that for many sequences an optimal balance between the two previous problems can be obtained.

Example. The frequency of an event in a sequence may show variation at different levels. Figure 1 shows an example of the relative frequency of an event over time, which is computed using two incremental sliding windows of lengths 1562 and 6250. The generative process for this sequence is described in Section 5.2. We observe that each window length tells us a different "story" about the frequency of the event. In other words, each window describes a different view of the data: the longer window suggests a "smoothly" increasing frequency throughout the sequence, while the shorter window captures a periodic behavior in the event frequency.

Contribution. In this paper we introduce the novel problem of finding a good set of window lengths for analyzing discrete sequences. We define suitable criteria and an efficient method for choosing window lengths, and give examples of tasks that demonstrate the applicability of the problem to different domains. We perform extensive experiments on real data from two application domains: text books and DNA sequences. We find that the scales of the occurrence patterns of various events (e.g., word types or DNA segments) vary significantly, and that the optimal scales can provide useful new insight into the data. Finally, we conduct an evaluation of optimal window lengths for random data to compare the empirical results with.

2 Related Work

String and Text Mining. Sliding windows have been used extensively in string mining. Indexing methods for string matching based on *n-grams* [21], i.e., subsequences of length n, employ sliding windows of fixed or variable length to create dictionaries and speed-up approximate string search in large collections of texts. Determining the appropriate window length is always a challenge, as small window lengths result in higher recall but large index structures. In text mining, looking at different linguistic dimensions of text results in extracting different "views" of the underlying text structure [3]. One way to quantify these views is by using sliding windows. Recently, an interactive text analysis tool[1] has been developed for exploring the effect of window length on three commonly used linguistic measures: type-token ratio[2], proportion of hapax legomena, and average word length. However, the window length is user-defined.

Bioinformatics. Several sliding window approaches have been proposed for analyzing large genomes and genetic associations. Two groups of methods exist in the literature that are characterized by fixed-length and variable-length sliding windows [4,22,26,29,32]. For the case of fixed-length windows it is hard to determine the optimal window length per task while variable-length windows provide higher flexibility. A variable window length framework for genetic association analysis employs principal component analysis to find the optimum window length [31]. Sliding windows have also been used for searching large biological sequences for poly-regions [29], motifs [7], and tandem repeats [2]. Nonetheless, in all cases mentioned above it is assumed that there exists only one optimum length and the solution is limited to the task of genetic association analysis.

Stream Mining. A common task in stream mining is to detect and monitor frequent items or itemsets in an evolving stream, counted over sliding windows. We present a brief survey of the use of sliding windows in stream mining, even the overall setting is very different from the problem studied in this paper and a setup requiring online learning is not covered by this paper. In the case of the fixed-length window model the length of the window is set at the beginning, and the data mining task is to discover recent trends in the data contained in the window [8,13,15,17]. In the time-fading model [20] the full stream is taken into account in order to compute itemset frequencies while time sensitivity is emphasized so that recent transactions get a higher weight as compared to earlier transactions. In addition, a tilted-time window [12] can be seen as a combination of different

[1] http://www.uta.fi/sis/tauchi/virg/projects/dammoc/tve.html

[2] The ratio of distinct tokens (words) to the total number of tokens in the text.

scales reflecting the alteration of the time scales of the windows over time. In the landmark model, particular time periods are fixed while the landmark designates the start of the system until the current time [16,17]. A frequency measure based on a flexible window length was introduced [5], where the frequency of an item is defined as the maximal frequency over all windows until the most recent event in the stream. Several variants of the above methods have been proposed, as well as adaptations of basic methods for different objectives.

Time Series. A common data mining problem in time series is the enumeration of previously unknown but frequently occurring patterns. Such patterns are called "motifs" due to their close analogy to their discrete counterparts in computational biology. Efficient motif discovery algorithms have been proposed, based on sliding windows, for summarizing and visualizing massive time series databases [6,27]. A method for discovering locally optimal patterns in time series at multiple scales has been proposed [28] along with a criterion for choosing the best window lengths. This is, however, a local heuristic and applies only to continuous data.

Based on the above discussion, sliding windows have been widely used in many application domains that involve sequences (discrete or continuous). However, window lengths are chosen either empirically or they are optimized for the task at hand. To the best of our knowledge, no earlier work has proposed a principled method for choosing the set of appropriate window lengths that optimally summarize the data for a given statistic and data mining task.

3 Problem Setting

3.1 Preliminaries

Given a set of event labels σ, a *sequence* of events is defined as $X = x_1 \dots x_n$, with each $x_t \in \sigma$. We denote as $X_j(i) = x_i \dots x_{i+j-1}$ the *subsequence* of length j starting at position i in X. We quantify the "information" of a subsequence $X_j(i)$ with a *statistic* $f(X_j(i))$. For example, f may be defined as the relative frequency of an event $q \in \sigma$ in $X_j(i)$, i.e.,

$$f(X_j(i)) = \frac{\#\ of\ occurrences\ of\ q\ in\ X_j(i)}{|X_j(i)|}, \tag{1}$$

where, by definition, $|X_j(i)| = j$. Alternatively, f may be defined as the type/token ratio of a sequence, i.e.,

$$f(X_j(i)) = \frac{\#\ of\ distinct\ events\ in\ X_j(i)}{|X_j(i)|}. \tag{2}$$

In principle f can be any function, but in the experiments (Sections 5 and 6) we use only the two functions given in Equations (1) and (2).

Since X may be structured at different levels with respect to statistic f, we are interested in finding the set of k window lengths that capture most of the structure in X. A window is defined as a slice of a sequence [25], or in other words it corresponds to a subsequence of X.

3.2 Problem Definition

Our goal is to capture several different levels of structure with respect to f by optimizing an objective function. Depending on the task at hand one may consider different objective functions. The objective function we propose in this paper (described in Problem 1 below) is to explain most of the "variation" in the data.

Let Ω be the set of all window lengths that we would like to consider in analyzing the structure of X. We also assume that we are given a distance function $d(\omega_i, \omega_j)$ that, given two window lengths $\omega_i, \omega_j \in \Omega$, quantifies the distance between the two structures in X captured by each of them. We present suitable choices for d in Section 4, an example is the sum of squared errors.

We propose to find the k window lengths that capture most of the variance in X:

Problem 1 (Maximal variance). Given a discrete sequence X, find a set $\mathcal{R} = \{\omega_1, \ldots, \omega_k\}$ of k window lengths that explain most of the variation in X, i.e., find a set $\mathcal{R}$ that minimizes

$$\sum_{\omega_i \in \Omega} \min_{\omega_j \in \mathcal{R}} d(\omega_i, \omega_j).$$

The above formulation corresponds to clustering. The resulting window lengths will be the centroids of the k clusters that explain the variation in X at different levels, and together the k centroids explain most information present in all possible window lengths.

The centroids can be viewed as code-book vectors that could be used to present the time series with any window length, with a minimal quadratic loss. Similar techniques are used, e.g., in lossy image compression to quantize the color space, where the code-book vectors can then be used to represent the pixels in fewer number of bits [10].

The above formulation is useful and applicable to real data scenarios, as shown by our experimental findings in Section 6. A method for solving Problem 1 is discussed in the following section.

4 WinMiner

In this section, we describe our proposed method, called `WinMiner`. We first introduce an auxiliary data structure, called *Window-Trace matrix*. Then, we describe an algorithm for solving Problem 1 using this matrix. We also study the computational complexity of `WinMiner` and present a sampling approach for reducing the time and space complexity of the method.

4.1 The Window-Trace Matrix

To solve Problem 1 we use an auxiliary matrix, called the *Window-Trace (W-T) matrix*, which is used to store the values of statistic f for each sliding window

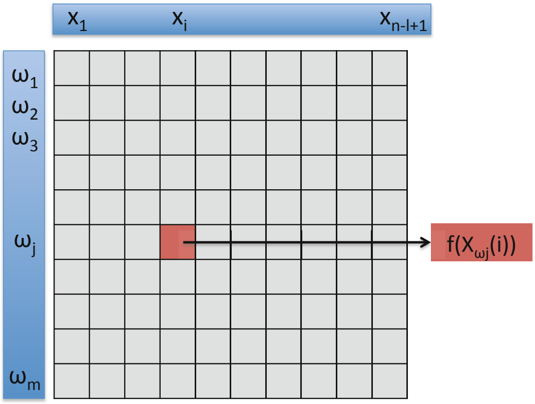


Fig. 2. Illustration of a W-T matrix $\mathcal{T}$ for function f, given a sequence $X = x_1 \ldots x_n$ and a set of window lengths $\Omega = \{\omega_1, \ldots, \omega_m\}$. Each cell in the matrix corresponds to the value of f at position i in X, for window length ω_j.

in X. More specifically, let X be the input sequence and f the statistic at hand. Then the W-T matrix $\mathcal{T}$ contains all values of $f(X_j(i))$ for all window lengths and a restricted set of sequence positions. We define $l = \arg\max_{\omega_i} \omega_i \in \Omega$ and $m = |\Omega|$. Then, $\mathcal{T}$ is given by

$$\mathcal{T}_{ji} = f(X_{\omega_j}(i)) \quad \forall i \in 1, \ldots, m, j \in 1, \ldots, n - l + 1. \tag{3}$$

Effectively, row i of matrix $\mathcal{T}$, denoted as $\mathcal{T}_{i*}$, contains a time series describing the behavior of statistic f over time for window length ω_i. In this setting, $d(\omega_i, \omega_j)$ can be defined to express the distance between the corresponding time series of ω_i and ω_j, i.e., rows $\mathcal{T}_{i*}$ and $\mathcal{T}_{j*}$. In our experiments, d is set to be the sum of squared errors. An illustration of $\mathcal{T}$ is shown in Figure 2.

4.2 WinMiner

The minimization problem described in Problem 1 is equivalent to clustering. According to our problem setting, a set of k representative window lengths needs to be identified.

We use $d(\omega_i, \omega_j) = \sum_{k=1}^{n-l+1} (\mathcal{T}_{ik} - \mathcal{T}_{jk})^2$, i.e., the sum of squared errors, as the distance function between two rows of $\mathcal{T}$, which results in Problem 1 being equivalent to the k-means problem. In general, the k-means problem is NP-Hard, but in practice we can use the iterative k-means algorithm to obtain a good approximation efficiently.

The execution of the k-means algorithm on $\mathcal{T}$ results in k clusters of window lengths. However, the centroids of these clusters do not necessarily correspond to a single window length, and in practice the centroids will often be a weighted sum of many window lengths. We obtain the final *approximately optimal* set of window lengths $\mathcal{R}$ by choosing the k window lengths that correspond to rows of $\mathcal{T}$ that have the smallest distance to each of the k cluster centroids. Note that K-means is run *rep* times and the best solution is reported. We call this algorithm `WinMiner` and its pseudocode is given in Algorithm 1.

Algorithm 1. Finding the k most variant points WinMiner(d, Ω, k, *rep*)

```
for i = 1 to $rep$ do
    $\{C_1, \ldots, C_k\}$ = K-means$(d, k)$ {k-means returns $k$ cluster centroids.}
    $\mathcal{R}_i = \{\}$
    for j = 1 to $k$ do
        $\mathcal{R}_i = \mathcal{R}_i \cup \{\arg\min_{r \in \Omega} d(C_j, r)\}$
    end for
    $loss_i = \sum_{\omega \in \Omega} \min_{r \in \mathcal{R}_i} d(\omega, r)$
end for
$best = \arg\min_{i \in \{1,\ldots,rep\}} loss_i$
return $R_{best}$
```

4.3 Computational Complexity

Let n be the size of the data and $m = |\Omega|$, as in Section 4.1. We then have that the size of the Window-Trace matrix $\mathcal{T}$ is $\mathcal{O}(m \cdot n)$. Also, the computational complexity and memory required to create and store it are equal to the size. The computational complexity for `WinMiner` is then the number of repetitions times the complexity of K-means over the matrix $\mathcal{T}$. Each iteration of K-means starts with an expectation step, in which each of the m points of dimension n is compared to each of the k cluster centroids, and then assigns them to the closest. In the ordinary k-means algorithm, the maximization step takes only m times n steps, because each point belongs to only one cluster. Thus, assuming the algorithm requires i iterations to converge, the total complexity of `WinMiner` is $\mathcal{O}(rep \cdot i \cdot k \cdot n \cdot m)$.

In the experiments in Sections 5 and 6, K-means is limited to 200 iterations, but often much less iterations are required to reach convergence. In the next section we discuss that there is usually no need to compute $\mathcal{T}$ over the entire data set and sampling can be used to greatly reduce the complexity of `WinMiner`.

4.4 Reducing the Complexity by Sampling

As shown in Section 4.3, one factor that affects the time complexity of `WinMiner` is the number of columns of $\mathcal{T}$. We can speed up our algorithm by sampling uniformly a small set of columns from $\mathcal{T}$, instead of using the full matrix.

In Section 5.2, we investigated empirically what would be the appropriate sampling rate to obtain a solution close to the solution that was obtained on the full matrix, assuming that the underlying sequence was generated using a variable rate Bernoulli process.

5 Evaluation on Synthetic Data

In any data mining task, it is important to be able to evaluate the significance of a result. Because of the complex set-up of our method, it is difficult to derive analytical results for what to expect regarding optimal sets of window lengths, expected cost of clustering, or expected minimal distances for the furthest pairs algorithm, even for simple random processes such as the Bernoulli process. To provide a baseline for the results in Section 6, we designed five experiments based on randomly generated data, where we know precisely what the properties of the data are.

5.1 Bernoulli Process with Fixed Rate

We are interested in the performance of two statistics: the cost of the optimal clustering and the set of window lengths given by `WinMiner`. We can use Algorithm 2 to generate random data from a Bernoulli process with fixed rate, given parameters n and p, which are the length of the sequence and the probability of the event occurring at any position, respectively.

Algorithm 2. Simulate a fixed-rate Bernoulli process SIM1(n, p)

```
for $i = 1$ to $n$ do
    $X(i) = Bernoulli(p)$
end for
```

Experiment 1. Since `WinMiner` is a non-deterministic approximation algorithm, the output may vary, even with the same input sequence. In the first experiment, we tested the stability of the solution in terms of the optimal window lengths given by `WinMiner`. Because `WinMiner` returns a set, comparing two solutions is not trivial. For brevity and ease of interpretation, we use $k = 3$ and only one sequence generated by Algorithm 2. We use $n = 10{,}000$ and $p = 0.1$ to generate the sequence and the number of repetitions (parameter *rep*) for `WinMiner` is set to 5, which should ensure reasonable approximations. The statistic is set to the relative frequency of the event, as defined in Equation 1. The results are presented in Figure 3. We observe that the result of `WinMiner` is the same in 97 out of 100 repetitions. The stability of the solution indicates that the chosen number of repetitions (parameter $rep = 5$) is sufficiently high.

Experiment 2. A data set, even with the same parameter settings, may give quite different results. Thus, secondly, we tested the stability of the solutions given by `WinMiner` for various values of k. We generated 100 data sets and tested the optimal window lengths for $k = 3, \ldots, 5$ on each data set. The other parameters were kept the same as in the previous experiment. The results are presented in Figure 3. We observe there is much more variation than in the previous experiment, which can be explained by the fact that a different input

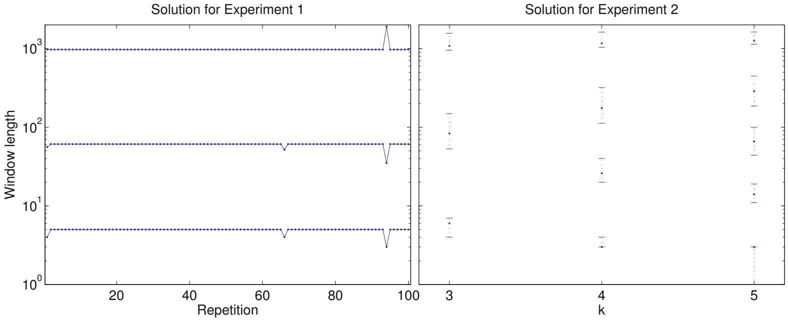


Fig. 3. Solutions for Experiments 1 and 2. The left figure shows the solutions found by `WinMiner` over repeated runs on a single sequence, using $k = 3$. We observe that the solutions are very stable over the different runs, which indicates that the chosen number of repetitions is sufficient. The right figure shows the solutions found by `WinMiner` over 100 synthetic data sets generated using fixed parameters, for increasing values of k. The solid dots represent the median values and the dashed lines give the 90% confidence intervals. We observe that there is more variance in this case, and the amount can be used as a baseline for other experiments.

sequence is used for each repetition. The observed variance in the figure can be used in future experiments to draw conclusions with respect to the significance of differences in sets of window lengths obtained for various events or data sets.

Experiment 3. Thirdly, we tested how the solutions depend on the average frequency of an event in the sequence. We leave most of the parameters as in the previous experiment, but now produce only one solution for each problem and repeat the process for varying value of p. We vary p from 0.01 to 0.50 in steps of 0.01. We do not have to study the behavior for $p > 0.50$ because the results will be symmetric to those between $p = 0.01$ and $p = 0.50$. The results are presented in Figure 4 and they suggest that there is no clear pattern. Hence, we can conclude that the event frequency has no direct influence on the optimal window lengths.

5.2 Bernoulli Process with Variable Rate

In the previous experiments, the frequency of the event remained fixed over time, which leads to the sequence having structure only on a single scale. To test the power of `WinMiner` in finding the true underlying scale at which the data is structured, we designed an algorithm to simulate a Bernoulli process with variable rate.

The full process is described in Algorithm 3. The first component of the variable rate is based on a slow increase of the event frequency over time, which ranges from $0.5 \cdot p$ at the start to $1.5 \cdot p$ at the end of the sequence X. The second component consists of the event frequency going up and down rhythmically, based on a sine wave with peak amplitude 0.5 and mean 0. Finally, both components are added together to give the variable event frequency, multiplied by the parameter p. The extra parameter, c, decides the periodicity of the sine

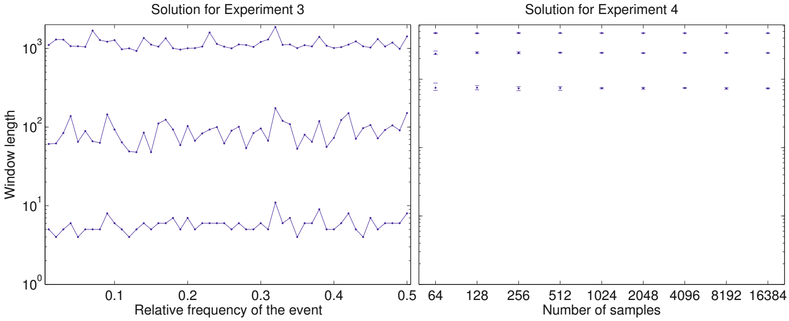


Fig. 4. Solutions for Experiments 3 and 4. The left figure shows the solutions found by `WinMiner` over sequences obtained from simulating a Bernoulli process with fixed rate p, for varying values of p. We find that there is no direct correlation between the optimal set of window lengths and the event frequency. The right figure shows the solutions found by `WinMiner` on a sequence obtained from simulating a Bernoulli process with rate that varies over time, using various numbers of samples. The solid dots represent the median values and the dashed lines give the 90% confidence intervals. We observe that the solutions for 1,024 and more samples are practically equivalent.

wave, and thus the second scale. We have generated a sequence with parameters n = 100,000, $p = 0.1$ and $c = 16$. The sequence has 10,009 events and has also been used to generate Figure 1.

Algorithm 3. Simulate a variable-rate Bernoulli process SIM2(n, p, c)

```
for $i = 1$ to $n$ do
    $t_1 = 0.5 + (i-1)/(n-1)$; // Multiplier for scale 1: [0.5–1.5]
    $t_2 = 0.5 \cdot \sin(c \cdot 2 \cdot \pi \cdot (i-1)/(n-1))$; // Multiplier for scale 2: [−0.5–0.5]
    $X(i) = Bernoulli(p \cdot (t_1 + t_2))$
end for
```

Experiment 4. As discussed in Section 4.4, we can obtain the optimal set of window lengths for this sequence without analyzing the full W-T matrix. We investigated empirically how many samples of $\mathcal{T}$ we would need (by performing uniform sampling on the columns of $\mathcal{T}$) to obtain a solution close to the solution that was obtained on the full matrix, i.e., the solution in Figure 5. We have varied the number of samples from 64 to 16,384 using powers of 2 and computed the solution 10 times for each sample size to assess the variance. We have used window lengths from 1 up to $\lfloor n/c \rfloor$ = 6,250 (which is the scale of the second component in the data) and $k = 3$. Figure 4 illustrates the results for `WinMiner`. We observe that the solution for Problem 1 is remarkably robust; the solutions using only 64 samples are already quite accurate approximations and from 1024 samples and up, the solutions are practically equivalent. Thus, we can conclude that 1,024 samples is sufficient for this data.

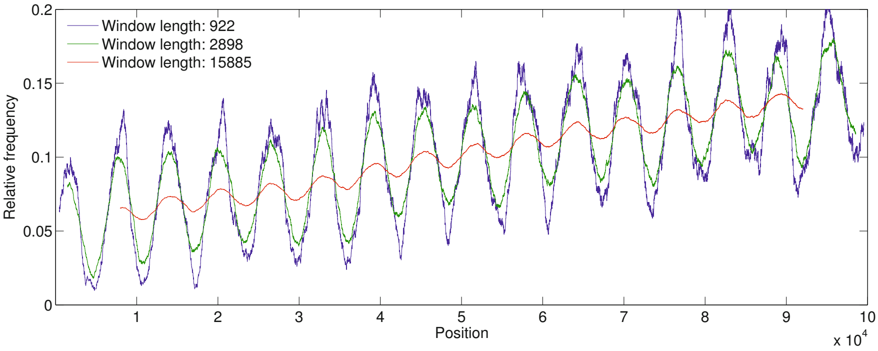


Fig. 5. The representation of the data based on the solution for $k = 3$ of Problem 1 on a sequence obtained from simulating a Bernoulli process with rate that varies over time. The variable trend in the data is clearly shown by the shorter window lengths, while the longest window length reveals the slow trend.

Experiment 5. Finally, we tested if we can retrieve the two scales that are present in the synthetic sequence. To prevent making it too easy for the algorithm, we use window lengths up to 20,000 and generate only 1,000 columns of the Window-Trace matrix $\mathcal{T}$. In a typical setting, we do not know how many scales a data set has. It is useful to note that a higher k always provides more information, thus choosing k too high is better than too low. For exploratory purposes, we use $k = 3$. In previous experiments we found that the solution for Problem 2 always includes the smallest window length, thus, to obtain a more interpretable result, the minimum window length is set to 50. Figure 5 illustrates the results for `WinMiner`. We find that the variable trend in the data can be identified well by solving Problem 1.

6 Evaluation on Real Data

To evaluate the usefulness of our problem setting in practice, we have designed three experiments on real data. In Section 6.1 we consider tracking the frequency of several words of varying type and frequency throughout the novel *Pride and Prejudice*. In Section 6.2 we study what window lengths would be appropriate for tracking the evolution of type/token ratio throughout several novels of *Charles Dickens* and try to relate the findings to previous linguistic research. Finally, in Section 6.3, we examine tracking the frequency of nucleotides and pairs of nucleotides in two reference genomes from the NCBI repository.

6.1 Optimal Window Lengths for Several Words

The influence of *burstiness* [18] and *dispersion* [14] of words in natural language corpora has become an important concept in research in linguistics [14], natural

Table 1. Using the MLE estimate for the Weibull β parameter as a measure of burstiness, we have selected these 24 words for comparison in our experiments. The words are the four most and least bursty words in three manually chosen frequency brackets. Bursty words exhibit greater variation in local frequency and non-bursty words are almost equally frequent throughout the book.

Frequency	Non-bursty	Index	Bursty	Index
Low [39–41]	met, rest, right, help	1–4	write, de, william, read	5–8
Medium [175–228]	time, soon, other, only	9–12	lady, has, can, may	13–16
High [600–1666]	with, not, that, but	17–20	you, is, my, his	21–24

language processing [24] and text mining [23]. Burstiness and dispersion are both indicators for the stability of the frequency of a word, i.e., a poorly dispersed or very bursty word tends to be highly frequent in some (parts of) texts and infrequent in all other (parts of) texts. The difference between the two measures is the level of granularity used in the analysis; burstiness is computed over running text, while dispersion is measured at the level of texts. In Section 5.1, we concluded that the optimal set of window lengths does not have a relation to the frequency of the event studied, thus it would be interesting to know if the optimal set of window lengths does depend on the burstiness of an event in a sequence.

To test this, we used the following experiment. We downloaded the popular novel *Pride and Prejudice* by Jane Austen, which is freely available through Project Gutenberg[3]. The novel has approximately 120,000 words. We then selected 24 words from three frequency bins, of which 12 are bursty and 12 are non-bursty. In this case, we measured the burstiness of a word by fitting a Weibull distribution to the inter-arrival time distribution of the word, then the shape parameter of the distribution is a measure for burstiness [1,23]. The Weibull (or stretched exponential) distribution is a two-parameter exponential family distribution which can be used to model the distribution of the interarrival-times of the words. The words are listed in Table 1. To study the effect of burstiness on the optimal sets of window lengths, we varied the parameter k from three to five and used window lengths from 1 to 4000.

The result is shown in Figure 6. The measurements for the bursty words are highlighted using a gray background. The results for Problem 1 (blue lines) are very interesting. We observe that for the non-bursty words, the results indeed appear to be all the same. Interestingly, the sets of optimal window lengths clearly contain longer windows for the bursty words, for any choice of k. This may be due to the fact that the bursty word exhibits a larger scale structure (bursts and intervals between bursts) than non-bursty more uniformly distributed words.

6.2 Type/Token Ratio throughout Several Novels

A recent study in linguistics considered the homogeneity of 14 novels by Charles Dickens [9]. We investigated if the optimal set of window lengths shows significant

[3] http://www.gutenberg.org/

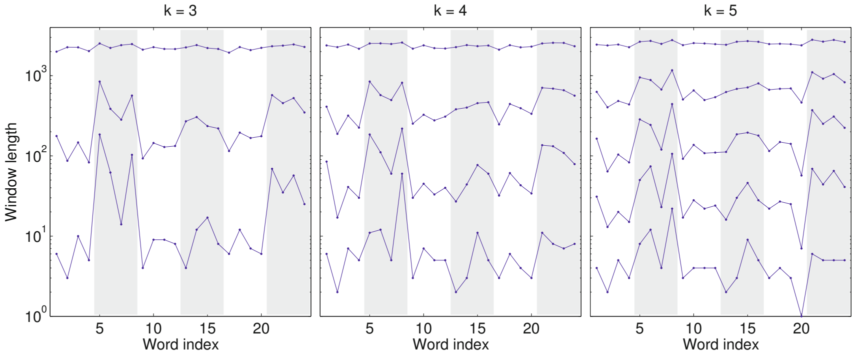


Fig. 6. Optimal sets of window lengths for analyzing the evolving frequency of 24 words in the novel *Pride and Prejudice*, for various choices of k. Each dot corresponds to a window length selected for that word, and lines are added to aid the comparison over words. The words range from low to high frequency and have the same order as in Table 1. We observe that the window lengths depend on the burstiness of a word, e.g., the fifth to eight word have high burstiness and have consistently higher window lengths than words one to four, which are not bursty. The same holds for words 13 to 16 and 21 to 24, which are also bursty, compared to words 9 to 12 and 17 to 20.

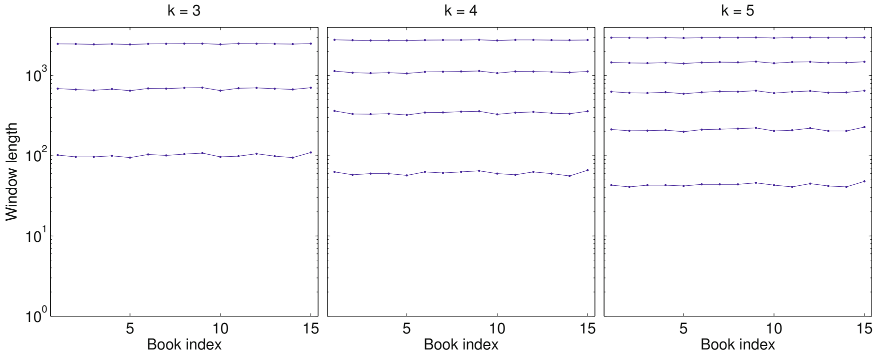


Fig. 7. Optimal sets of window lengths for analyzing the evolving type/token ratio of 15 novels, for various choices of k. Each dot corresponds to a window length selected for that word, given the value of k, and lines are added to aid the comparison over words. Surprisingly, the solution set for each of the novels is almost the same.

variation over this set of novels. We downloaded the fourteen novels discussed by Evert [9], and used Pride and Prejudice as the fifteenth novel. We again used window lengths from one to 4,000 and varied k from three to five. However, the statistic used is now type/token ratio, as given in Equation (2).

The result is shown in Figure 7. The sets of window lengths found by our method is remarkably robust, suggesting that the novels are indeed quite similar in structure.

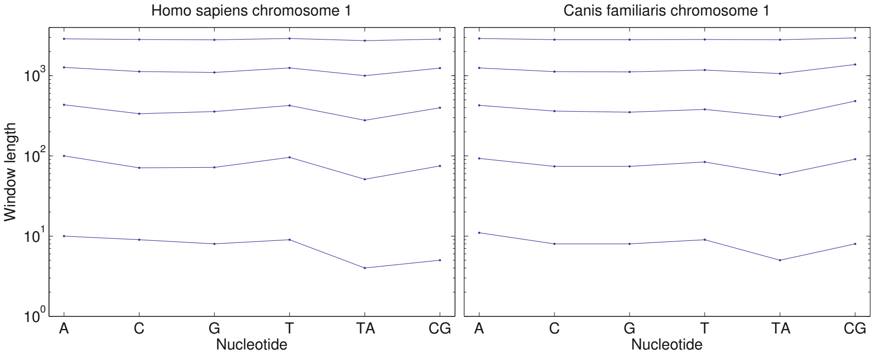


Fig. 8. Optimal sets of window lengths for analyzing the evolving frequency of nucleotides in Homo sapiens chromosome 1 and Canis familiaris chromosome 1, for $k = 5$

6.3 Frequency of Nucleotides in Biological Sequences

Studies in biology and bioinformatics have shown that DNA chains consist of a number of important, known functional regions, at both large and small scales, which contain a high occurrence of one or more nucleotides [29]. Examples of such regions include: *isochores*, which correspond to multi-megabase regions of genomic sequences that are specifically GC-rich or GC-poor and exhibit greater gene density; *CpG islands*, that correspond to regions of several hundred nucleotides that are rich in the dinucleotide CpG which is generally under-represented (relative to overall GC content) in eukaryotic genomes and there presence in the genome has been associated with gene expression in nearby genes.

We have studied Chromosome 1 of two organisms: `Homo sapiens` (human) and `Canis familiaris` (dog), of lengths 225 and 122 million nucleotides, respectively. The data has been downloaded from the NCBI data repository[4]. We focused on six event types: the four nucleotides A, C, G, and T, as well as dinucleotides TA and CG. We tested `WinMiner` for $k = 5$ and window lengths up to 4,000. The statistic used in our experiments was the relative event frequency. To speed up our method we sampled 10,000 columns from the W-T matrix.

In Figure 8 we see a comparison of the 5 best window lengths found by `WinMiner` for the two organisms. We observe that the four single nucleotides as well as dinucleotide CG (which is indicative of the gene structure) exhibit highly similar behavior for both organisms. This is explained by the high genomic structural similarity between humans and dogs [19]. Nonetheless, we see that dinucleotide TA has a substantially different behavior. This is due to the fact that TA is known to be the least stable of all dinucleotides [11] and in many cases indicative of Transcription Factor Binding Sites (TFBS), which are different in the two organisms.

[4] `http://www.ncbi.nlm.nih.gov`

7 Conclusions

We have studied the novel problem of identifying a set of window lengths that contain the maximal amount of information in the data. We have presented a generally applicable objective function that users could employ, which can also be efficiently optimized. We have extensively studied the performance of the proposed optimization algorithm, as well as the identified solutions for two examples of sliding window statistics on both synthetic data and real data. We have illustrated that the results on synthetic data are useful as a baseline for practical use, and that sampling can be used to obtain the optimal set of window lengths based on a small sample of the data, making the method practical for (collections of) sequences of any size. Moreover, we have shown that the window lengths themselves can show interesting properties of the data; among other findings, we have identified the relation between the burstiness of events and the optimal window lengths.

An open problem is how many window lengths users should employ in practical settings. In principle, more windows is always more informative, and in many situations users may be able to easily identify when the set of results is too large.

References

1. Altmann, E.G., Pierrehumbert, J.B., Motter, A.E.: Beyond word frequency: Bursts, lulls, and scaling in the temporal distributions of words. PLoS ONE 4(11), e7678 (2009)
2. Benson, G.: Tandem repeats finder: a program to analyze dna sequences. Nucleic Acids Research 27(2), 573–580 (1999)
3. Biber, D.: Variation across speech and writing. Cambridge University Press (1988)
4. Bourgain, C., Genin, E., Quesneville, H., Clerget-Daproux, F.: Search for multifactorial disease susceptibility genes in founder populations. Annals of Human Genetics 64(03), 255–265 (2000)
5. Calders, T., Dexters, N., Goethals, B.: Mining frequent items in a stream using flexible windows. Intelligent Data Analysis 12(3), 293–304 (2008)
6. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: Proc. of ACM SIGKDD, pp. 493–498 (2003)
7. Das, M.K., Dai, H.-K.: A survey of DNA motif finding algorithms. BMC Bioinformatics 8(suppl. 7), S21 (2007)
8. Demaine, E.D., López-Ortiz, A., Munro, J.I.: Frequency Estimation of Internet Packet Streams with Limited Space. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 348–360. Springer, Heidelberg (2002)
9. Evert, S.: How random is a corpus? the library metaphor. Zeitschrift für Anglistik und Amerikanistik 54(2), 177–190 (2006)
10. Forsyth, D., Ponce, J.: Computer Vision: A Modern Approach. Prentice Hall (2011)
11. Gentles, A.J., Karlin, S.: Genome-scale compositional comparisons in eukaryotes. Genome Research 11(4), 540–546 (2001)
12. Giannella, C., Han, E.R.J., Liu, C.: Mining frequent itemsets over arbitrary time intervals in data streams. Technical Report TR587 (2003)
13. Golab, L., López-Ortiz, A., Dehaan, D., Munro, J.I.: Identifying frequent items in sliding windows over on-line packet streams. In: Proc. of IMC, pp. 173–178 (2003)

14. Gries, S.T.: Dispersions and adjusted frequencies in corpora. International Journal of Corpus Linguistics 13(4), 403–437 (2008)
15. Jin, L., Chai, D.J., Lee, Y.K., Ryu, K.H.: Mining frequent itemsets over data streams with multiple time-sensitive sliding windows. In: Proc. of ALPIT, pp. 486–491 (2007)
16. Jin, R., Agrawal, G.: An algorithm for in-core frequent itemset mining on streaming data. In: Proc. of IEEE ICDM, pp. 210–217 (2005)
17. Karp, R.M., Shenker, S., Papadimitriou, C.H.: A simple algorithm for finding frequent elements in streams and bags. ACM Trans. Database Syst. 28(1), 51–55 (2003)
18. Katz, S.M.: Distribution of content words and phrases in text and language modelling. Natural Language Engineering 2(1), 15–59 (1996)
19. Kirkness, E.F., Bafna, V., Halpern, A.L., Levy, S., Remington, K., Rusch, D.B., Delcher, A.L., Pop, M., Wang, W., Fraser, C.M., Venter, J.C.: The dog genome: survey sequencing and comparative analysis. Science 301(5641), 1898–1903 (2003)
20. Lee, D., Lee, W.: Finding maximal frequent itemsets over online data streams adaptively. In: Proc. of IEEE ICDM, pp. 266–273 (2005)
21. Li, C., Wang, B., Yang, X.: Vgram: improving performance of approximate queries on string collections using variable-length grams. In: Proc. of VLDB, pp. 303–314 (2007)
22. Li, Y., Sung, W.-K., Liu, J.J.: Association mapping via regularized regression analysis of single-nucleotidepolymorphism haplotypes in variable-sized sliding windows. The American Journal of Human Genetics 80(4), 705–715 (2007)
23. Lijffijt, J., Papapetrou, P., Puolamäki, K., Mannila, H.: Analyzing Word Frequencies in Large Text Corpora Using Inter-arrival Times and Bootstrapping. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 341–357. Springer, Heidelberg (2011)
24. Madsen, R.E., Kauchak, D., Elkan, C.: Modeling word burstiness using the dirichlet distribution. In: Proc. of ICML, pp. 545–552 (2005)
25. Mannila, H., Toivonen, H., Inkeri Verkamo, A.: Discovery of frequent episodes in event sequences. Data Min. Knowl. Discov. 1(3), 259–289 (1997)
26. Mathias, R., Gao, P., Goldstein, J., Wilson, A., Pugh, E., Furbert-Harris, P., Dunston, G., Malveaux, F., Togias, A., Barnes, K., Beaty, T., Huang, S.-K.: A graphical assessment of p-values from sliding window haplotype tests of association to identify asthma susceptibility loci on chromosome 11q. BMC Genetics 7(1) (2006)
27. Mueen, A., Keogh, E.J., Zhu, Q., Cash, S., Westover, B.: Exact discovery of time series motifs. In: Proc. of SIAM SDM (2009)
28. Papadimitriou, S., Yu, P.: Optimal multi-scale patterns in time series streams. In: Proc. of ACM SIGMOD, pp. 647–658 (2006)
29. Papapetrou, P., Benson, G., Kollios, G.: Discovering frequent poly-regions in dna sequences. In: Proc. of IEEE ICDM Workshops, pp. 94–98 (2006)
30. Sörnmo, L., Laguna, P.: Bioelectrical Signal Processing in Cardiac and Neurological Applications. Elsevier Academic Press (2005)
31. Tang, R., Feng, T., Sha, Q., Zhang, S.: A variable-sized sliding-window approach for genetic association studies via principal component analysis. Annals of Human Genetics 73(Pt 6), 631–637 (2009)
32. Toivonen, H., Onkamo, P., Vasko, K., Ollikainen, V., Sevon, P., Mannila, H., Herr, M., Kere, J.: Data mining applied to linkage disequilibrium mapping. Am. J. Hum. Genet. 67, 133–145 (2000)

Discovering Links among Social Networks

Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino

DIMET, University "Mediterranea" of Reggio Calabria, Via Graziella,
Località Feo di Vito, 89122 Reggio Calabria, Italy
{bucca,lax,a.nocera,ursino}@unirc.it

Abstract. Distinct social networks are interconnected via bridge users, who play thus a key role when crossing information is investigated in the context of Social Internetworking analysis. Unfortunately, not always users make their role of *bridge* explicit by specifying the so-called `me` edge (i.e., the edge connecting the accounts of the same user in two distinct social networks), missing thus a potentially very useful information. As a consequence, discovering missing `me` edges is an important problem to face in this context yet not so far investigated. In this paper, we propose a common-neighbors approach to detecting missing `me` edges, which returns good results in real life settings. Indeed, an experimental campaign shows both that the state-of-the-art common-neighbors approaches cannot be effectively applied to our problem and, conversely, that our approach returns precise and complete results.

Keywords: Link Prediction, Link Mining, Social networks, Social Internetworking.

1 Introduction

In the last years (on-line) social networks have been showing an enormous development becoming probably the main actor of the Web 2.0. The rapid and revolutionary diffusion of social networks among all segments of the population has attracted the interest of many researchers from disparate fields [19], such as sociology, psychology, economy, computer science, etc., also for the applications which the analysis of involved data can enable. In this landscape, Social Network Analysis and Social Network Mining [9] have assumed an important role, since both the hugeness of data and their graph-based organization have enforced the development of specific models and methods allowing the study of social-network data to discover knowledge from them. Clearly, the graph-based data schema gives a great information power to links among data, since it allows people profiles, resources, activities, and so on, to be directly (and indirectly) related. The crucial role of relationships in the expression of an individual's personality as well as in her social identity, traditionally recognized by social sciences, is even strengthened in the field of virtual societies, where relationship links are the main form of expression of participation of individuals to the community. To make more challenging the analysis of this reality it concurs the fact

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 467–482, 2012.

that the scenario where we operate is not the one of single, isolated, independent social networks, but a universe composed of a constellation of several social networks, each forming a community with specific connotations, but strongly interconnected with each other. It is a matter of fact that, despite the inherent underlying heterogeneity, the interaction among distinct social networks is the basis of a new emergent internetworking scenario enabling a lot of strategic applications, whose main strength will be just the integration of possibly different communities yet preserving their diversity and autonomy. Clearly, social mining and analysis approaches may strongly rely on this huge multi-network source of information, which also reflects multiple aspects of people personal life, thus enabling a lot of powerful discovering activities.

From this perspective, links among different social networks assume a fundamental role. They typically connect the same user among the social networks she belongs to, and derive from the explicit user's declaration (sometimes supported and encouraged by specific tools) consisting in the insertion of `me` edges [1]. Unfortunately, for disparate reasons, not always users make their role of *bridge* explicit by specifying the `me` edge, missing thus a potentially very useful information. As a consequence, we may view the overall underlying (social internetworking) graph as a graph where a big number of missed `me` edges exists, whose discovery thus represents a very important issue. In other words, an interesting problem of missing link detection arises, which partially overlaps with a link prediction issue, since we may expect that a portion of missing `me` edges will be inserted in a next stage in the graph.

In this paper, we deal with the above problem by proposing an effective solution experimentally tested in a real-life Social Internetworking Scenario (SIS, for short). To the best of our knowledge, the problem of detecting `me` edges has not been investigated in the literature, but the approach we adopt in this work, which exploits a recursive notion of common-neighbor similarity, suggested us to prior verify whether common-neighbor approaches for link prediction [21] can be directly applied to our problem. The answer to this question was definitely negative, as intuitively explained in Section 2 and experimentally confirmed in Section 5, thus motivating our work. Our solution is thus based on a notion of node similarity, whose exploitation allows us to detect whether a suitable threshold is exceeded and then a missing `me` edge between two nodes is detected. The similarity between two nodes is obtained by combining two contributions: a string similarity between the associated user names, and a contribution based on a suitable recursive notion of common-neighbor similarity. The latter is extremely important because, in the literature, it is well known that string similarity alone can lead to synonymy and homonymy errors [24,15]; the neighborhood similarity allows these errors to be detected and avoided. In order to motivate the above choice it is important to clarify that the problem we are addressing does not deal with the case in which a user voluntarily keeps two accounts separated in their respective social networks. In this case, she chooses account names very different from each other, she does not have common friends and, very probably, one of the two profiles is fake (i.e., it does not contain real information about her). In this case any approach to detecting node similarity would presumably fail. This situation, which is closer to identity-management and security problems, is little relevant in our context, where we are interested in completing the real profile of users. For these users we may expect (and this is just what we found for me-connected accounts) that a user joining two (or more) social networks tends to have at least partially overlapping sets of friends in them. Therefore, neighbors are useful information to exploit in order to detect missing me edges.

The plan of this paper is as follows: in the next section, we examine related literature. In Section 3, we present our recursive notion of similarity. On the basis of this notion, we design the method we use to detect missing me edges. This is described in Section 4. In Section 5, we illustrate the experiments we have carried out to verify the performances of our technique. Finally, in Section 6, we draw our conclusions and sketch possible future evolutions of our research.

2 Related Work

The detection of me edges in a SIS can be seen as a special case of the problem of identifying users on the Web. As a matter of fact, it allows the features of bridge users to be detected. Identifying users on the Web has received a great attention in several application scenarios, such as personalization. A lot of work is devoted to verify whether user profile information can be sufficient to address this problem. In [13] the authors define and implement a framework that provides a common base for user identification for cross-system personalisation among Web-based user-adaptive systems. The corresponding user identification algorithm combines a set of identification properties, such as username, name, location or email address, and classifies a user as identified if such a combination exceeds a suitable threshold. In [18], a technique based on user profiles for identifying users across social systems is proposed. This technique has been successfully validated on three social tagging networks (Flickr, Delicious and StumbleUpon). The limit of this technique is that only few users make their profile available in social tagging platforms. A method to identify users on the basis of profile matching is proposed in [26]. In this paper data from two popular social networks are used to evaluate the importance of fields in the Web profile and to develop a profile comparison tool. The authors of [29] provide evidence on the existence of mappings among usernames across different communities. Starting from the observation of the data in BlogCatalog, they infer 7 hypotheses on the relationships among the usernames selected by a single person in different communities. On the basis of such hypotheses, they propose an approach that, given a username u in a source community and a target community c, generates a set of candidate usernames in c corresponding to u. The approach first generates a set of usernames from u by adding and removing suitable prefixes and suffixes. Then, it exploits a Web search on Google aimed at checking for the existence of each candidate username in such a way as to reduce the returned set of usernames.

From another point of view, the detection of me edges in a SIS is someway related to link prediction. Link prediction is a task of link mining aiming at predicting the (even future) existence of a link between two objects [21,6]. In the contest of social networks, it focuses on predicting friendships among users. Often, social networks are represented as graphs [11]. As a consequence, some link prediction approaches are totally based on the structural properties of these graphs [20]. A first possibility to perform this task consists in analyzing common neighbors. In order to decide whether two nodes are related, [5] exploits a similarity measure derived from the Jaccard coefficient. Based on *preferential attachment* [23], [8] experimentally verifies that the probability of a relationship between two nodes is proportional to the product of the number of their neighbors. Some approaches to link prediction rely on the notion of shortest-path distance which is computed by means of several similarity measures, like the Katz coefficient, PageRank and SimRank. Due to the high computational cost of these measures, some approximations have to be adopted in order to make them effective. In any case, whenever the number of nodes is considerable, the application of these methods may result in a too long running time. In conjunction with all the above techniques, some strategies may be used to enhance the accuracy of predictions. Also the use of *unseen bigrams* [14] can help in the link detection task. Here, the similarity between a node A and a node B is computed by taking into account the similarity between the nodes B and C, where this last one is the node most similar to A. Furthermore, the quality of link detection can be improved by means of clustering techniques aiming at identifying the graph components which introduce noise in the similarity computation [20]. [25] proposes the application of statistical relational learning to link prediction in the domain of scientific literature citations. In this approach statistical modeling and feature selection are integrated into a search mechanism over the space of database queries in such a way as to define feature candidates involving complex interactions among objects in a given relational database. [27] analyzes the localization in space and time of a large number of users by means of their call detail records. This analysis shows that users with similar movement routines are strongly connected in a social network and have intense direct interactions. This result allows implicit ties in the social network to be predicted with a significant accuracy starting from the analysis of the correlation between user movements (i.e., their mobile homophily). Other approaches to link detection come from the fields of *deduplication* and *disambiguation.* In particular, [7] proposes an algorithm for discovering duplicates in the dimensional tables of a Data Warehouse.

From the above analysis it emerges that our approach, in the above literature, can be related only with common-neighbors ones. However, despite their apparent closeness to ours, we can easily realize that they are not directly applicable to our context. Indeed, the notion of common-neighbors relies, in general, on the notion of common identity of the friends of a user. But discovering the common identity of users in different social networks is for us the output of the problem, leading to a sort of recursive definition of the problem itself. We have experimentally confirmed the above claim by showing that the application of the state-of-the-art common-neighbors approaches to our problem returns very unsatisfying results. The results of these experiments are reported in Section 5.

3 The Notion of Similarity

Our approach operates in a Social Internetworking System (SIS) resulting from the interconnection of a number of distinct social networks. We start with the basic notion of underlying graph, which is the layer we deal with.

Definition 1. A *t-Social-Internetworking Graph (SIG)* is a directed graph $G = \langle N, E \rangle$, where N is the set of *nodes*, E is the set of *edges* (i.e., ordered pairs of nodes) and N is partitioned into t subsets $S_1, \ldots, S_t$. Given a node $a \in N$ we denote by $S(a)$ the social graph which a belongs to. E is partitioned into two subsets E_f and E_m. E_f is said the set of *friendship edges* and E_m is the set of `me` *edges*. E_f is such that for each $(a, b) \in E_f$, $S(a) = S(b)$, while E_m is such that for each $(a, b) \in E_m$, $S(a) \neq S(b)$[1]. Given a node a we denote by $\mathit{\Gamma}(a)$ the set of nodes in $S(a)$ such that for each $b \in \mathit{\Gamma}(a)$ $(a, b) \in E_f$. $\mathit{\Gamma}(a)$ is said the set of *neighbors* of a. The graphs corresponding to $S_1, \cdots, S_t$ are called *social graphs* and in *SIG* they are linked to each other by means of `me` edges. □

A t-Social-Internetworking Graph $G = \langle N, E_f \cup E_m \rangle$ is the graph underlying a SIS composed of t social networks. Each node a of G is associated with a user, joining the social network whose underlying graph is $S(a)$. An edge $(a, b) \in E_f$ means that the user b is a friend of the user a in the social network of $S(a)$. An edge $(c, c') \in E_m$ means that the user c in the social network of $S(c)$ has declared a `me` edge between herself and the user c' in the social network of $S(c')$. In other words, c is a bridge and this means that c and c' are associated to the same user. From now on, throughout this section, consider given a t-Social-Internetwork-ing Graph $G = \langle N, E_f \cup E_m \rangle$.

Our approach is based on a recursive notion of "inter-social-network" similarity aimed at detecting missing bridges of G. The similarity between two nodes a, b (belonging to two different social networks) is obtained by combining two contributions: a string similarity between the user names associated to a and b, and a contribution based on a suitable notion of common-neighbors similarity. The latter component leads to a recursive definition of the overall inter-social-network similarity since the common-neighbors notion has to necessarily rely on the same notion, because also neighbors belong to different social networks, and, thus, common nodes have to be detected too. Observe that this two-component philosophy has been successfully adopted in several application fields in the past; among these fields we cite schema matching [24,15], information retrieval [10], and logic programming [17].

Concerning the string similarity, several functions have been proposed in the literature, such as Jaro-Winkler, Levenshtein, QGrams, Monge-Elkan, Soundex

[1] Observe that the presence of E_m makes a t-Social-Internetworking Graph not to be a forest since, in this last case, the corresponding social networks should be disjoint.

[16]. One of them could be adopted to measure user name similarity in our approach, so that it can be considered parametric w.r.t. the string-similarity function. We have evaluated the application of the different functions in Section 5.2.

Before defining our notion of similarity, we have to introduce a preliminary notion, which the common-neighbors contribution relies on. Indeed, we detect a missing `me` edge between a and b if a suitable combination of the string similarity between the user names associated to them, according to the metric Q, and the (recursive) similarity of the top-k similar pairs each composed of a friend of a and a friend of b, is greater than a suitable threshold, for a given k.

Thus, we have preliminarily to define how to select such top-k pairs. This is related to the next definition.

Definition 2. Given a positive integer k_0, a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, and a non-negative integer n we inductively define $Top_Q^n(a, b, k_0)$ as follows:

1. $Top_Q^0(a, b, k_0)$ is any subset of $C = \{(x^a, y^b) \mid x^a \in \Gamma(a), y^b \in \Gamma(b)\}$ containing the top-k_0 elements of C w.r.t. the metric Q.
2. For any $0 < i \leq n$, $Top_Q^i(a, b, k_0)$ is any subset of $C = \{(x^z, y^w) \mid (z, w) \in Top_Q^{i-1}(a, b, k_0), x \in \Gamma(z), y \in \Gamma(w), (x^*, *) \notin Top_Q^j(a, b, k_0), (*, y^*) \notin Top_Q^j$ $(a, b, k_0), 0 \leq j \leq i-1\}$ containing the top-k_i elements of C w.r.t. the metric Q, where $k_i = \lceil \frac{k_0}{(1+i)^{1+i}} \rceil$ and $*$ denotes any node in N. □

Concerning the contribution of neighbors, we have to consider a particular situation which could significantly affect the precision of our technique. Suppose we have two nodes z and w belonging to different social networks and consider $x \in \Gamma(z)$ and $y \in \Gamma(w)$. If x and y are power users (i.e., users having a very high degree in social networks) and z and w are not, it could happen that x (resp., y) belongs to $\Gamma(z)$ (resp., $\Gamma(w)$) only because it corresponds to a public figure (e.g., a V.I.P.). In this case, the presence of this node in $\Gamma(z)$ and $\Gamma(w)$ is not significant in defining the real life relationships of z and w. In order to prevent this effect, we introduce the following reduction coefficient $\gamma(x^z, y^w)$, which assumes a very powerful role in the above situation.

Definition 3. Let $x, y, z, w \in N$ be nodes of G such that $x \in \Gamma(z)$, $y \in \Gamma(w)$, and $S(z) \neq S(w)$. We define: $\gamma(x^z, y^w) = min(\delta(x^z), \delta(y^w))$ where $\delta(a^b) = \frac{max(|\Gamma(a)|, |\Gamma(b)|)}{max(|\Gamma(a)|, |\Gamma(b)|) + ||\Gamma(a)| - |\Gamma(b)||}$ for any pair of nodes $a, b \in N$. □

Now we are ready to define our similarity function. As said above, it is obtained as a combination of two contributions, namely, the string-similarity component and the common-neighbors one. The underlying intuition is that if two accounts, belonging to different social networks, are associated to the same user, even though there is no `me` edge between them, they will have user names someway related each other and, moreover, they share a (even low) number of friends. It is worth remarking that this condition, easily verifiable by exploring real-life social networks, does not mean that the two users have a strong overlap of their neighbors, coherently to the diversity among the communities included in different social networks. However, we argue that it is highly probable that an even little neighbor overlap will occur[2].

Definition 4. Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, two integers $n \geq 0$ and $k_0 > 0$, we inductively define the *similarity operator* $T_Q^n(a, b, k_0)$ as follows:

1. $T_Q^0(a, b, k_0) = Q(a, b)$.
2. For any $0 < i \leq n$,

$$T_Q^i(a, b, k_0) = (1 - \beta_i) \cdot T_Q^{i-1}(a, b, k_0) + \beta_i \cdot \frac{\sum_{(x^z, y^w) \in Top^i_{\widetilde{Q}}(a,b,k_0)} \widetilde{Q}(x^z, y^w)}{|Top^i_{\widetilde{Q}}(a,b,k_0)|}$$

where $\beta_i = \frac{1}{(i+1)^{i+1}}$ and $\widetilde{Q}(x^z, y^w) = \gamma(x^z, y^w) \cdot Q(x, y)$, for any $x, y, z, w \in N$ nodes of G such that $x \in \Gamma(z)$, $y \in \Gamma(w)$, and $S(z) \neq S(w)$. □

The definition of similarity is recursive. At the basic step, only the direct string-similarity value concurs to define the similarity between two nodes a and b. At step i, the similarity is obtained as a linear combination of the similarity of the step $i-1$, and the new common-neighbors contribution. This is obtained as the average of the reduced (by γ – see Definition 3) string similarity between the top-k_i pairs w.r.t. the same metric. k_i is derived, at each step, as an exponential reduction of k_0, which is an input parameter allowing us to modulate the size of the neighbors overlapping considered relevant for the similarity computation. Observe that the above linear combination depends on the β_i parameter, which is exponentially decreasing as i increases, making quickly less important the common-neighbors contribution when leaving from the root nodes a and b.

Now we are ready to define the effective tool we provide to detect `me` edges, obviously based on the above notion of similarity. Indeed, Definition 4 would lead to an ineffective computation covering, in the worst case, the whole graph G. Anyway, we can observe that when, during the computation, we reach a step h whose contribution to the overall similarity is under a given small ϵ, we expect that from now on involved neighbors do not give us any meaningful information. Thus, we can stop here the iteration. This is encoded into the next definition.

Definition 5. Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, an integer number $k_0 > 0$, and a real number $\epsilon > 0$, we define the ϵ-*similarity* $S_Q^\epsilon(a, b, k_0)$ between a and b w.r.t. Q as $T_Q^h(a, b, k_0)$, where $h > 0$ is the least number (if any) such that $|T_Q^h(a, b, k_0) - T_Q^{h-1}(a, b, k_0)| < \epsilon$. □

Clearly, our approach is really effective if the ϵ-similarity $S_Q^\epsilon(a, b, k_0)$ between two nodes a and b always exists. This is ensured by the following theorem.

[2] Recall that we are not interested in cases of users who voluntarily keep two accounts separated in their respective social networks, where the above conditions are not verified, as explained in the Introduction.

Theorem 1. *Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, an integer number $k_0 > 0$, and a real number $\epsilon > 0$, then the ϵ-similarity $S^{\epsilon}_{Q}(a, b, k_0)$ between a and b w.r.t. Q exists.* □

Finally, in the next theorem we give a result related the complexity of our technique, showing that it is feasible.

Theorem 2. *Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, and an integer number $k_0 > 0$, then the number of visited nodes in G for the computation of $T^{h}_{Q}(a, b, k_0)$ is $O(d^2 \cdot k_0)$, where d is the maximum node degree.* □

The consequence of Theorem 2 is that, despite the potentially exponential explosion of visited nodes for the computation of the ϵ-similarity $S^{\epsilon}_{Q}(a, b, k_0)$ between two nodes a and b, due to the h-step iterative navigation of neighbors in G, the number of visited nodes is independent of the number of iterations h and is bounded by the number of visited nodes at the first two steps. As it can be easily verified by reading the proof of the theorem, this fact clearly depends on how k_i is defined in Definition 2. The above result ensures the feasibility of our method.

4 `Me`-Edge Detection

In this section, we present our method able to discover missing `me` edges, and, thus, new links among different social networks. Clearly, these links complement the `me` edges explicitly declared by users. Our technique is based on the notion of similarity presented in Section 3. In particular, the similarity function allows us to detect a missing `me` edge between two nodes, whether a suitable threshold is exceeded. We consider given a Social Internetworking System (SIS) composed of t social networks, as well as the underlying Social Internetworking Graph $G = \langle N, E_f \cup E_m \rangle$.

Since, in a SIS, the number of node pairs to consider for the possible presence of a `me` edge is enormous, we have identified a mechanism leading our approach to consider only a reasonable number of very promising pairs. In particular, from the examination of the explicitly declared `me` edges, we have found that, with a high probability, some of the nodes belonging to the neighbors of two nodes linked by a `me` edge are, in their turn, linked by a `me` edge. As a consequence, our approach starts from a set of already known `me` edges and examines only the neighbors of the nodes involved in these edges. Clearly, if this set of `me` edges is not available, our approach can work all the same by starting from any pair of nodes in G. Thus, our algorithm receives a set $M \subseteq E_m$ of existing `me` edges and returns a set M' of discovered `me` edges, such that, clearly $M' \cap (E_f \cup E_m) = \emptyset$. The function *detectMe* implementing our approach is reported in Algorithm 1. It receives as arguments: the set M of starting `me` edges, two [0,1] real variables th_c and th_d, an integer $k_0 > 0$, and (a small) real $\epsilon > 0$. Recall that the similarity between two nodes a, b defined in Section 3 is obtained by combining two contributions: a string similarity between the user names associated to

a and b, and a contribution based on a suitable notion of common-neighbors similarity. Concerning the first contribution, as pointed out in Section 3, there exist several already defined functions for computing the similarity between two strings, each characterized by specific features (e.g., Jaro-Winkler, Levenshtein, QGrams, Monge-Elkan, Soundex, etc. [16]). The function $Q(a, b)$ (receiving two nodes a and b) in Algorithm 1 can be considered parametric w.r.t. the string-similarity function. We can choose one of the above functions on the basis of the desired target. For instance, QGrams is very severe and assigns quite low similarity degrees, Jaro-Winkler is more permissive whereas Soundex is very permissive. In Section 5.2 the application of the different functions in our technique is experimentally evaluated.

Synthetically, Algorithm 1 proceeds as follows. For each edge (a, b) in M, we take all pairs (a', b') such that $a' \in \Gamma(a)$ and $b' \in \Gamma(b)$. These pairs are candidate `me` edges. A pair (a', b') is discarded if the value of $Q(a', b')$ is lower than a suitable threshold th_c or if $(a', b') \in E_m$. Otherwise, a function S is called, which implements the computation of $S_Q^{\epsilon}(a', b', k_0)$ (see Definitions 4 and 5). If the value returned by this function is greater than a suitable threshold th_d, then (a', b') is detected as `me` edge and is inserted in M'. Otherwise, it is discarded. The function S is reported in Algorithm 2. It is recursive since it implements also the operator $T_Q^n(a, b, k_0)$ of Definition 5. It receives as arguments: an integer $k_0 > 0$, a (small) real $\epsilon > 0$, a list L of triplets $\langle a, b, s\rangle$, where a and b are nodes and s is a $[0, 1]$ real value, the value of S at the previous step (at the initial step of the recursion this value coincides with $Q(a, b)$), and an integer $i > 0$ representing the step of the recursion. To explain what is s, observe that, in order to implement $T_Q^n(a, b, k_0)$, we need as argument the list L which stores, at the step $i > 1$ of the recursion, the top-k_i node pairs w.r.t. the metric $\widetilde{Q}^i(a, b, k_0)$. Thus, in this case, s represents just $\widetilde{Q}^i(a, b, k_0)$. At the initial step of the recursion ($i = 1$), s is just the string similarity value $Q(a, b)$.

The correspondence between Algorithm 2 and Definitions 4 and 5 is quite clear. We just have to highlight that the result of the computation of the operator $Top_{\widetilde{Q}}^i(a, b, k_0)$ (see Definitions 2 and 5) is embedded in the list L, as described above.

The computation of $Top_{\widetilde{Q}}^i(a, b, k_0)$ is implemented by Algorithm 3. The function Top receives two nodes a and b and a positive integer k_i. It returns a list L of k_i triplets $\langle a', b', \widetilde{Q}^i(a', b', k_0)\rangle$, where $a' \in \Gamma(a)$, $b' \in \Gamma(b)$ and (a', b') is one of the top-k_i pairs w.r.t. the metric $\widetilde{Q}$. We remark that the metric here used includes the reduction factor γ of Definition 3.

5 Experiments

In this section we present our experimental campaign aimed at determining the performances of our approach. Since it operates on a SIS, we had to extract not only the connections among the accounts of different users in the same social network but also the connections among the accounts of the same user in

Algorithm 1. *detectMe*

```
Input $M$: the starting set of me edges
Input $th_c$: A candidate threshold
Input $th_d$: A detection threshold
Input $k_0$: an integer
Input $\epsilon$: a real
Output $M'$: the set of detected me edges
Variable $L$: a list of triplets of the form $\langle a, b, s \rangle$
$L := \emptyset$; $M' := \emptyset$
for each edge $(a,b) \in M$ do
    for each node pair $(a',b') \in \Gamma(a) \times \Gamma(b)$ do
        if ($Q(a',b') > th_c$ and the edge $(a',b') \notin E_m$) then
            insert the triplet $\langle a', b', Q(a',b') \rangle$ into $L$
            if ($S(k_0, \epsilon, L, Q(a',b'), 1) > th_d$) then
                insert the me edge $(a',b')$ in $M'$
            end if
            $L := \emptyset$
        end if
    end for
end for
return $M'$
```

Algorithm 2. *S*

```
Input $k_0$: an integer
Input $\epsilon$: a real
Input $L_i$: a list of triplets $\langle a, b, s \rangle$
Input $S_{i-1}$: the similarity value of $a'$ and $b'$ at step $i-1$
Input $i$: an integer
Output $S_i$: the similarity value of $a'$ and $b'$ at step $i$
Variable $\beta$: a [0,1] real
Variable $k_i$: an integer
Variable $avgS$: a real
Variable $L_{i+1}$: a list of triplets $\langle a, b, s \rangle$
$L_{i+1} := \emptyset$; $\beta := \frac{1}{i^i}$; $k_i := \lceil k_0 \cdot \beta \rceil$
if (k=1) then
    $S_i := S_{i-1}$
else
    assign to $avgS$ the average value of the similarities of the node pairs of $L_i$
    $S_i := (1 - \beta) \cdot S_{i-1} + \beta \cdot avgS$
end if
if ($|S_i - S_{i-1}| \geq \epsilon$) then
    for each $\langle a, b, s \rangle$ in $L_i$ do
        add the list returned by $Top(a, b, k_i)$ to $L_{i+1}$
    end for
    return $S(i+1, L_{i+1}, S_i)$
else
    return $S_i$
end if
```

different social networks. In order to handle these connections, two standards encoding human relationships are generally exploited. The former is XFN (XHTML Friends Network) [3]. It simply uses an attribute, called `rel`, to specify the kind of relationship between two users. Some possible values of `rel` are `me`, `friend`, `contact`, `co-worker`, `parent`, and so on. A (presumably) more complex alternative to XFN is FOAF (Friend-Of-A-Friend) [2]. In both of them information about `me` edges is the one explicitly declared by users. In our experiments, we considered a SIS consisting of four social networks, namely Twitter, LiveJournal, YouTube and Flickr. Therefore, from now on we refer to a (real-life) *t*-Social

Algorithm 3. *Top*

```
Input $a, b$: a node
Input $k_i$: an integer
Output $L$: a list of triplets $\langle a, b, s \rangle$
Variable $t$: a triplet $\langle a, b, s \rangle$
Variable $c$: an integer
Variable $\delta_a, \delta_b, \widetilde{Q}_{(a,b)}$: a real
$L := \emptyset$; $c := 0$
for each $a'$ in $\Gamma(a)$ do
    for each $b'$ in $\Gamma(b)$ do
        $\delta_a := \frac{max(|\Gamma(a')|,|\Gamma(a)|)}{max(|\Gamma(a')|,|\Gamma(a)|)+||\Gamma(a')|-|\Gamma(a)||}$
        $\delta_b := \frac{max(|\Gamma(b')|,|\Gamma(b)|)}{max(|\Gamma(b')|,|\Gamma(b)|)+||\Gamma(b')|-|\Gamma(b)||}$
        $\widetilde{Q}_{(a,b)} := min(\delta_a, \delta_b) * Q(a', b')$
        if $(c < k_i)$ then
            insert the triplet $\langle a', b', \widetilde{Q}_{(a,b)} \rangle$ into $L$
            sort $L$ in a descending order
            $c := c + 1$
        else
            $t := L.get(k_i - 1)$
            if $(\widetilde{Q}_{(a,b)} > t.s))$ then
                replace the triplet in the position $(k_i - 1)$ of $L$ with the triplet $\langle a', b', \widetilde{Q}_{(a,b)} \rangle$
                sort $L$ in a descending order
            end if
        end if
    end for
end for
return $L$
```

Internetworking Graph such that $t = 4$. We argue that the relatively small number of involved social networks, as a first investigation, is adequate for our purpose, expecting that the results we obtain in this setting are still valid in a more complex environment. Anyway, the above social networks are highly representative (they are among the top-10 social networks in terms of population). As a matter of fact, they are largely analyzed in the past in Social Network Analysis [22,28].

For our experiments, we used a server equipped with a 2 Quad-Core E5440 processor and 16 GB of RAM with the CentOS 6.0 Server operating system. We performed our experiments from January 30, 2012 to April 5, 2012.

5.1 Application of the State-of-the-Art Common-Neighbors Approaches

As described in Section 2, we have to prior verify whether common-neighbor approaches for link prediction [21] can be directly applied to our problem. However, a first aspect has to be considered. In our scenario the notion of common neighbors cannot be the classical one, because the neighbors of examined pairs belong to different social networks. As a consequence, we cannot expect that two examined neighbors have some common nodes in strict sense. To overcome this drawback we have just to re-define the notion of node identity. Coherently with our setting, it simply suffices to consider as *identical* two nodes linked by

Table 1. The tested common-neighbors approaches

Index Name	*Definition*	*Sensitivity*
Salton Index (SAI)	$s_{ab}^{SAI} = \frac{\lvert \Gamma(a) \bigcap \Gamma(b) \rvert}{\sqrt{\lvert \Gamma(a) \rvert \times \lvert \Gamma(b) \rvert}}$	0.01
Jaccard Index (JAI)	$s_{ab}^{JAI} = \frac{\lvert \Gamma(a) \bigcap \Gamma(b) \rvert}{\lvert \Gamma(a) \bigcap \Gamma(b) \rvert}$	0.01
Sorensen Index (SOI)	$s_{ab}^{SOI} = \frac{2 \lvert \Gamma(a) \bigcap \Gamma(b) \rvert}{\lvert \Gamma(a) \rvert + \lvert \Gamma(b) \rvert}$	0.01
Hub Promoted Index (HPI)	$s_{ab}^{HPI} = \frac{\lvert \Gamma(a) \bigcap \Gamma(b) \rvert}{min(\lvert \Gamma(a) \rvert, \lvert \Gamma(b) \rvert)}$	0.00
Hub Depressed Index (HDI)	$s_{ab}^{HDI} = \frac{\lvert \Gamma(a) \bigcap \Gamma(b) \rvert}{max(\lvert \Gamma(a) \rvert, \lvert \Gamma(b) \rvert)}$	0.01
Leicht-Holme-Newman Index (LHNI)	$s_{ab}^{LHNI} = \frac{\lvert \Gamma(a) \bigcap \Gamma(b) \rvert}{\lvert \Gamma(a) \rvert \times \lvert \Gamma(b) \rvert}$	0.01
Resource Allocation Index (RA)	$s_{ab}^{RA} = \sum_{z \in \Gamma(a) \bigcap \Gamma(b)} \frac{1}{\lvert \Gamma(z) \rvert}$	0.01
Local Path Index (LPI)	$s_{ab}^{LPI} = A^2 + \epsilon A^3$ (A is G's adjacency matrix)	0.03

a `me` edge. At this point, classical common-neighbor techniques can be directly applied.

Given two nodes a and b in G (such that $S(a) \neq S(b)$), the considered (state-of-the-art) techniques are those reported in Table 1, where we include, in the first and second columns, the definition of the similarity index which they rely on [21].

We tested all the above techniques in our SIS by preliminarily constructing a set M of 100 node pairs linked by a `me` edge and then by running them on M. For each technique, we obtained a set M' of detected `me` edges. Clearly, M' represents a set of true positives. Finally, we measured the *sensitivity* of the techniques as the ratio $\frac{|M'|}{|M|}$, obtaining the results reported in the third column of Table 1.

The analysis of such values clearly shows that no effective result can be obtained whether common-neighbors techniques are adopted. This has in fact motivated our further study, whose experimental validation is reported in the next sections.

5.2 Sample-Driven Method Validation

A first experiment aims at determining the performance of our approach as well as at choosing the best function for computing the string similarity in our context. We started from the set M introduced in the previous section (i.e., a set of 100 real `me`-edge-connected pairs). Then, we find another set, denoted by $\neg M$, of 100 node pairs not connected by a `me` edge. To find two elements, say (c_1, d_1) and (c_2, d_2), of $\neg M$, we started from a `me`-edge-connected pair (a, b) and then we required that $c_1 = a$, $d_1 \in \Gamma(b)$, $c_2 = b$, $d_2 \in \Gamma(a)$. This way, both (c_1, d_1) and (c_2, d_2) are not linked by a `me`-edge, since d_1 is a friend of a user (i.e., b) who is identical to c_1. The dual situation occurs for (c_2, d_2).

Then, we applied our approach on the pairs of M and $\neg M$ and we obtained the sets TP, FP, and FN, which are true positives, false positives, and false negatives, resp. For clarity, an element in TP is a pair of M detected as `me`-edge-connected pair by our technique, an element in FP is a pair of $\neg M$ detected as

Table 2. Precision, recall and F-Measure of our approach for each user name similarity function

Function	*Precision*	*Recall*	*F-measure*
Jaro-Winkler	0.558	0.920	0.694
QGrams	0.908	0.690	0.784
Levenshtein	0.877	0.710	0.785
Smith-Waterman	0.840	0.790	0.814
Smith-Waterman-Gotoh	0.779	0.810	0.794
Monge-Elkan	0.779	0.810	0.794
Needleman-Wunch	0.500	1.000	0.667
Jaro	0.555	0.910	0.689
Soundex	0.500	0.990	0.664

`me`-edge-connected pair by our technique, and an element of FN is a pair of M not detected as `me`-edge-connected pair by our technique.

To compute the performance of our approach we adopted three classical measures [4], namely *precision* (as measure of correctness), *recall* (as measure of completeness) and *F-measure* (as the harmonic mean of precision and recall). They are defined as: $precision = \frac{|TP|}{|TP|+|FP|}$, $recall = \frac{|TP|}{|TP|+|FN|}$, and $F\text{-}measure = 2 \cdot \frac{precision \cdot recall}{precision+recall}$.

Since the behavior of our approach (and, consequently, the values of precision and recall) depends on the function adopted for computing string similarity, we considered the most common of these functions and, for each of them, we computed the precision and the recall of our technique. This way, we were able to determine the function(s) maximizing these measures. Obtained results are shown in Table 2.

The main conclusion we can draw from the analysis of this table is that our approach presents in general a very satisfying performance both in correctness and in completeness. Moreover, we observe that we are free to choose the string-similarity function in a rich set. Indeed, 5 functions led our approach to obtain a precision higher than 0.77 and 6 functions led it to obtain a recall higher than 0.81. However, among the considered functions, QGrams (resp., Needleman-Wunch) proved to be the one capable of assuring the best precision (resp., recall). The high performance level of our approach is even more evident if we compare Tables 1 and 2 and if we consider that, in this testbed, the definitions of recall and sensitivity coincide and, consequently, the corresponding columns can be compared[3].

5.3 Expert-Based Method Validation

This experiment aims at computing the accuracy of our approach in a way different from the previous experiment. In this case, we want to benefit from the support of a human expert. We first applied a crawling technique to derive a sample of the SIS. This sample was necessary to have a starting set of `me` edges at

[3] Observe that, owing to the extremely low values of sensitivity, the computation of precision in Table 1 makes no sense.

disposal. In order to maximize the number of me edges occurring in the sample we applied BDS, a crawling technique specifically conceived to operate on a SIS, instead of on a single social network, which is highly capable of finding and returning explicitly declared me edges [12] (note that, to the best of our knowledge, no other technique with this feature is available in literature). Our sample consisted of 93,169 nodes and 146,325 edges. 745 out of 146,325 were me edges. This sample can be found at the URL http://www.ursino.unirc.it/pkdd-12.html. We randomly selected 160 me edges and put them in a set M. We gave this set as input to our technique. The adopted string-similarity function was QGrams, because it proved to assure the best precision. Our approach returned a set M' of 22 me edges and a set of 133 non-me edges, from which we randomly selected a set $\neg M'$ of 22 non-me edges in such a way that me edges and non-me edges had the same weight. After this, we asked the human expert to verify whether the elements of M' are actually me edges and the elements of $\neg M'$ are actually non-me edges. For each edge her possible answers were *true*, *false* and *unknown*. Observe that the value *unknown* reflects both uncertain cases and unreachable-page ones. At the end of the experiment we obtained that, as for M', she returned $t_p = 16$ *true*, $f_p = 4$ *false* and 2 *unknown*. As for $\neg M'$, she returned $t_n = 18$ *true*, $f_n = 2$ *false* and 2 *unknown*. Finally, we computed the *accuracy* as the ratio $\frac{t_p+t_n}{t_p+t_f+t_n+f_n}$, obtaining the value 0.85, which denotes a very good performance of our method.

6 Conclusion and Future Work

In this paper, we have studied the problem of discovering missing me edges in a Social Internetworking Scenario. The most evident information we can use to detect missing me edges, besides user names, concerns neighbors. This might lead us to conclude that we are in front of a classical problem of link prediction where one of the state-of-the-art common-neighbors techniques can be applied in order to solve it. The first conclusion we have drawn in this work, on the basis of a number of experiments, is that this is definitely not true, showing the need of studying the problem as new and finding a specific solution. To do this, we have defined a suitable notion of "inter-social-network" similarity, which is recursive since the common-neighbors notion has to necessarily rely on the same notion, because also neighbors belong to different social networks. On the basis of this notion, we have defined an algorithm able to detect whether there is a missing me edge between two given nodes. The experimental analysis of this method on a real-life data set has shown its correctness and completeness, thus validating it. In the future, this analysis could be further empowered in several directions. Some of them could be: *(i)* the analysis of our approach running time; *(ii)* the analysis of the mutual role of the two components of our similarity definition; *(iii)* the analysis of the stability and sensitivity of our approach.

The results obtained in this paper can be useful for further investigations in the direction of social internetworking. Indeed, the role of me edges is relevant for any phenomenon of information crossing through different social networks, so that discovering new me edges may strongly enrich the analysis capabilities of social data, also strengthening multi-context analyses of people profile. We thus believe that a number of future directions of our research can be undertaken, especially in the context of behavioral analysis, but also at the "lower" level of graph analysis, where the discovery of new me edges may improve the capability of crawlers to explore a Social Internetworking Scenario.

Acknowledgment. This work was partially funded by the Italian Ministry of Research through the PRIN Project EASE (Entity Aware Search Engines).

References

1. Google Social Graph (2012), http://code.google.com/p/itswhoyouknow/wiki/SocialGraph
2. The Friend of a Friend (FOAF) project (2012), http://www.foaf-project.org/
3. XFN - XHTML Friends Network (2012), http://gmpg.org/xfn
4. ISO 5725-1:1994/Cor 1:1998. Accuracy (trueness and precision) of measurement methods and results - Part 1: General principles and definitions - Technical Corrigendum 1 (1998)
5. Adamic, L., Adar, E.: Friends and Neighbors on the Web. Social Networks 25(3), 211–230 (2003)
6. Al Hasan, M., Zaki, M.J.: A Survey of Link Prediction in Social Networks. In: Social Network Data Analytics, pp. 243–276. Elsevier (2011)
7. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in Data Warehouses. In: Proc. of the International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, China, pp. 586–597. VLDB Endowment (2002)
8. Barabási, A.L., Jeong, H., Neda, Z., Ravasz, E., Schubert, A., Vicsek, T.: Evolution of the social network of scientific collaborations. Physica A: Statistical Mechanics and its Applications 311(3-4), 590–614 (2002)
9. Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., Pedreschi, D.: As Time Goes by: Discovering Eras in Evolving Social Networks. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part I. LNCS, vol. 6118, pp. 81–90. Springer, Heidelberg (2010)
10. Bollegala, D.T.: A Study on Attributional and Relational Similarity between Word Pairs on the Web. PhD Thesis, University of Tokyo (2009)
11. Brocheler, M., Pugliese, A., Subrahmanian, V.S.: Probabilistic Subgraph Matching on Huge Social Networks. In: Proc. of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2011), Kahosiung, Taiwan, pp. 271–278 (2011)
12. Buccafurri, F., Lax, G., Nocera, A., Ursino, D.: Crawling Social Internetworking Systems. In: Proc. of the International Conference on Advances in Social Analysis and Mining (ASONAM 2012), Istanbul, Turkey (forthcoming, 2012)
13. Carmagnola, F., Cena, F.: User identification for cross-system personalisation. Information Sciences 179(1-2), 16–32 (2009)
14. Dagan, I., Pereira, F., Lee, L.: Similarity-based estimation of word cooccurrence probabilities. In: Proc. of the Annual Meeting on Association for Computational Linguistics, pp. 272–278. Association for Computational Linguistics (1994)

15. De Meo, P., Quattrone, G., Terracina, G., Ursino, D.: Integration of XML Schemas at various "severity" levels. Information Systems 31(6), 397–434 (2006)
16. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate Record Detection: A Survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)
17. Ferilli, S., Basile, T.M.A., Di Mauro, N., Biba, M., Esposito, F.: A New Similarity Measure for Guiding Generalizations Search. In: Proc. of the International Conference on Inductive Logic Programming, ICLP 2006, Seattle, Washington, USA (2006)
18. Iofciu, T., Fankhauser, P., Abel, F., Bischoff, K.: Identifying users across social tagging systems. In: Proc. of the International Conference on Weblogs and Social Media (ICWSM 2011), Barcelona, Catalonia, Spain. The AAAI Press (2011)
19. Kleinberg, J.: The convergence of social and technological networks. Communications of the ACM 51(11), 66–72 (2008)
20. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology 58(7), 1019–1031 (2007)
21. Lü, L., Zhou, T.: Link Prediction in Complex Networks: A Survey. Physica A: Statistical Mechanics and its Applications 390(6), 1150–1170 (2011)
22. Mislove, A., Koppula, H.S., Gummadi, K.P., Druschel, F., Bhattacharjee, B.: Growth of the Flickr Social Network. In: Proc. of the International Workshop on Online Social Networks (WOSN 2008), Seattle, Washington, USA, pp. 25–30. ACM (2008)
23. Newman, M.E.J.: The structure and function of complex networks. SIAM Review, 167–256 (2003)
24. Palopoli, L., Saccà, D., Terracina, G., Ursino, D.: Uniform techniques for deriving similarities of objects and subschemes in heterogeneous databases. IEEE Transactions on Knowledge and Data Engineering 15(2), 271–294 (2003)
25. Popescul, A., Ungar, L.H.: Statistical relational learning for link prediction. In: Proc. of the International Workshop on Learning Statistical Models from Relational Data, Acapulco, Mexico, vol. 149, pp. 172–179 (2003)
26. Vosecky, J., Hong, D., Shen, V.Y.: User identification across multiple social networks. In: Proc. of the International Conference on Networked Digital Technologies (NDT 2009), Ostrava, the Czech Republic, pp. 360–365 (2009)
27. Wang, D., Pedreschi, D., Song, C., Giannotti, F., Barabási, A.L.: Human mobility, social ties, and link prediction. In: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011), San Diego, California, pp. 1100–1108. ACM (2011)
28. Ye, S., Lang, J., Wu, F.: Crawling online social graphs. In: Proc. of the International Asia-Pacific Web Conference (APWeb 2010), Busan, Korea, pp. 236–242. IEEE (2010)
29. Zafarani, R., Liu, H.: Connecting corresponding identities across communities. In: Proc. of the International Conference on Weblogs and Social Media (ICWSM 2009), San Jose, California, USA. The AAAI Press (2009)

Efficient Bi-objective Team Formation in Social Networks

Mehdi Kargar, Aijun An, and Morteza Zihayat

Department of Computer Science and Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada, M3J 1P3
{kargar,aan,zihayatm}@cse.yorku.ca

Abstract. We tackle the problem of finding a team of experts from a social network to complete a project that requires a set of skills. The social network is modeled as a graph. A node in the graph represents an expert and has a weight representing the monetary cost for using the expert service. Two nodes in the graph can be connected and the weight on the edge represents the communication cost between the two corresponding experts. Given a project, our objective is to find a team of experts that covers all the required skills and also minimizes the communication cost as well as the personnel cost of the project. To minimize both of the objectives, we define a new combined cost function which is based on the linear combination of the objectives (i.e. communication and personnel costs). We show that the problem of minimizing the combined cost function is an NP-hard problem. Thus, one approximation algorithm is proposed to solve the problem. The proposed approximation algorithm is bounded and the approximation ratio of the algorithm is proved in the paper. Three heuristic algorithms based on different intuitions are also proposed for solving the problem. Extensive experiments on real datasets demonstrate the effectiveness and scalability of the proposed algorithms.

1 Introduction

Team formation has been traditionally studied in operational research. Most of the traditional approaches do not consider the network structure behind the individuals. Nowadays, various forms of online social networks have been developed, making it possible to consider the relationships among individuals when forming a team to complete a task or project. Recently, some works have been devoted to find a team of experts from a social network that minimizes the cost of communication among the experts [11,10]. Such a team should possess a set of skills in order to complete the task or project. While effective communication is indeed important for the success of a project, there are some other factors that are also important but have not been considered in team formation from social networks. One of these factors is the personnel cost. In reality, people normally need to be paid for working on a project, and it is desirable to find a team of experts with a reasonable personnel cost.

In this paper, we tackle the problem of finding the best team of experts that possess a set of skills while minimizing both the communication cost and the personnel

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 483–498, 2012.

cost. We use a graph to model a social network in which nodes represents experts, each having a set of skills and a cost for using his/her service. Two nodes in the graph can be connected and the weight on an edge represents the communication cost between the two connected experts. Generally, more previous collaborations between two experts, the lower the communication cost between them.

Clearly, our problem is a constrained bi-criteria optimization problem. A common approach to solving such a problem is to combine the two objectives into a single objective. In this paper, we take this approach and show that the problem of optimizing the combined objective function is NP-hard. Thus, we propose an approximation algorithm and three heuristic algorithms to efficiently solve the problem in polynomial time. The approximation algorithm is based on converting the input graph with both node and edge weights into a graph with weights on only edges. It has a performance guarantee with an approximation ratio of 2. The heuristic methods are based on either iteratively replacing cheapest experts with more expensive ones to improve the combined cost or incrementally adding experts with minimum cost contribution. We conduct extensive experiments on real data sets to show the effectiveness and efficiency of the proposed methods.

The paper is organized as follows. Related work is presented in Section 2. Problem statements and definitions are given in section 3. The algorithms for finding the best team of experts are presented in section 4. The experimental results are illustrated in section 5 and section 6 concludes the paper.

2 Related Work

Discovering a team of experts in a social network is introduced in [11]. The authors propose two functions for evaluating the communication cost among the members of a team. The communication cost functions are improved in [10]. The new function in [10] considers all the edges of the induced sub-graph, and is thus more stable to small or radical changes than the ones in [11]. The problem of finding a team of experts with a leader is also introduced in [10]. The problem is generalized by associating each required skill with a specific number of experts in [12]. The maximum load of the experts in the presence of several tasks is minimized in [1], but it does not consider finding teams with low communication cost. Recently, the problem of online team formation is studied in [2], which creates teams of experts with minimized work load and communication cost. The personnel cost of the experts is not considered in [2]. In this work, in addition to finding a team of experts with low communication cost, we also minimize the personnel cost of the team.

The problem of team formation has also been studied in the operation research community. Simulated annealing, branch and bound and genetic algorithms are used for solving the problem [4,13,14,7]. The main difference between this work and the works in operation research is that the experts are not connected through a social network in their work. The authors of [8], consider the effect of different graph structures among the members on the performance of the team. They performed their studies in an experimental setting and they do not study the problem from a computational point of view. Dynamics of group formation and its effect on the formation of groups in the network is considered in [3]. A game theoretic approach to this problem is discussed in [9]. Although these studies are not directly related to our setting, they might be considered as a complementary work for our problem.

The authors of [6] consider a bicriteria team formation problem. One objective is the communication cost and the other is the level of skills of the experts. It does not consider the personnel cost of the team. It produces a solution using a simulated annealing method and no approximation bound is given.

3 Problem Statement

Let $C = \{c_1, c_2, \ldots, c_m\}$ denote a set of m experts, and $S = \{s_1, s_2, \ldots, s_r\}$ denote a set of r skills. Each expert c_i has a set of skills, denoted as $Q(c_i)$, and $Q(c_i) \subseteq S$. If $s_j \in Q(c_i)$, expert c_i has skill s_j. In addition, a subset of experts $C' \subseteq C$ have skill s_j if at least one of them has s_j. For each skill s_j, the set of all experts having skill s_j is denoted as $C(s_j) = \{c_i | s_j \in Q(c_i)\}$. A project $P \subseteq S$ is defined as a set of skills which are required for the completion of the project. A subset of experts $C' \subseteq C$ is said to *cover* a project P if $\forall s_j \in P \; \exists \; c_i \in C', s_j \in Q(c_i)$.

The experts are connected together in a social network and it is modeled as an undirected and weighted graph (G). Each node in G represents an expert in C. Terms node and expert are used interchangeably through this paper. Two nodes are connected by an edge if the experts have collaborated before. The weight of an edge represents the communication cost between two experts. The lower the weight of the edge between two nodes, the more easily the two experts can collaborate or communicate, and the lower the communication cost between them. Each expert in the graph is also associated with a cost representing the monetary cost for using the expert service. The cost of an expert c_i is denoted as $t(c_i)$.

The **distance** between two nodes c_i and c_j, denoted as $d(c_i, c_j)$, is the sum of weights on the shortest path between them in G. It should be noted that the shortest distance function is a **metric** and satisfies the **triangle inequality**. If c_i and c_j are not connected in G (directly or indirectly), the distance between them is ∞.

Definition 1. ***(Team of Experts)*** *Given a set of experts C and a project P that requires a set of skills $\{s_1, s_2, \ldots, s_p\}$, a* team of experts *for P is a set of p skill-expert pairs: $\{\langle s_1, c_{s_1}\rangle, \langle s_2, c_{s_2}\rangle, \ldots, \langle s_p, c_{s_p}\rangle\}$, where c_{s_j} is an expert in C having skill s_j for $j = 1, \ldots, p$. A skill-expert pair $\langle s_i, c_{s_i}\rangle$ means that expert c_{s_i} is responsible for skill s_i in the project.*

Note that an expert may be responsible for more than one skill in a project. The same as our previous work [10], to evaluate the communication cost among the experts in a team T, we use the sum of distances among the experts of a team defined as follows.

Definition 2. ***(Sum of Distances)*** *Consider a graph G whose nodes represent experts and whose edges are weighted by the communication cost between two experts. Given a team T of experts from G for a project: $\{\langle s_1, c_{s_1}\rangle, \langle s_2, c_{s_2}\rangle, \ldots, \langle s_p, c_{s_p}\rangle\}$, the* sum of distances *of T with respect to G is defined as*

$$SD_G(T) = \sum_{i=1}^{p} \sum_{j=i+1}^{p} d(c_{s_i}, c_{s_j})$$

where $d(c_{s_i}, c_{s_j})$ is the distance between nodes c_{s_i} and c_{s_j} in G (as defined earlier).

To measure the personnel cost of a team, the following function is defined:

Definition 3. ***(Personnel Cost)*** *Consider a graph G whose nodes represent experts and each expert is associated with a cost. Given a team T of experts from G for a project: $\{\langle s_1, c_{s_1}\rangle, \langle s_2, c_{s_2}\rangle, \ldots, \langle s_p, c_{s_p}\rangle\}$, the* personnel cost *of T with respect to G is defined as*

$$PC_G(T) = \sum_{i=1}^{p} t(c_{s_i})$$

where $t(c_{s_i})$ is the cost of expert c_{s_i}.

In this cost model an expert x is paid $k \times t(x)$, where k is the number of skills the expert is responsible for in the project. This is reasonable because more skills the expert uses, more responsibility or tasks he/she has in the project.

We are interested in finding a team of experts that minimizes both the personnel and communication costs. Thus, our problem is a bi-objective optimization problem. One way to solve a bi-criteria optimization problem is to convert the problem into a single objective problem by combining the two objective functions into a single one. In this paper, we take this approach and define a single objective function that combines the communication and personnel costs with a tradeoff parameter (i.e., λ) as follows.

Definition 4. ***(Combined Cost Function)*** *Given a team T of experts from graph G for a project and a tradeoff λ between the communication and personnel costs, the combined cost of T with respect to G is defined as*

$$ComCost_G(T) = (p-1)(1-\lambda) \times PC_G(T) \; + \; 2\lambda \times SD_G(T)$$

where p is the number of required skills.

The parameter λ varies from 0 to 1 and indicates the tradeoff between the communication and personnel costs. Since the values of $PC_G(T)$ and $SD_G(T)$ may have different scales, $PC_G(T)$ and $SD_G(T)$ should be normalized before using the formula so that they both fall into the same range. The reason for having $(p-1)$ in the first term and 2 in the second is to scale up the terms because $PC_G(T)$ is the sum of p expert costs while $SD_G(T)$ is the sum of $\frac{p(p-1)}{2}$ pairwise communication costs. Given the combined cost function, we define the problem tackled in this paper as follows:

Problem 1. **(Team Formation by Minimizing the Combined Cost)** Given a project P, a graph G representing the social network of a set of experts and a tradeoff λ between the communication and personnel costs, the problem of team formation tackled in this paper is to find a team of experts T for P from G so that it covers all of the required skills and minimizes the combined cost function $ComCost_G(T)$ defined in 4.

Theorem 1. *Problem 1 is an NP-hard problem.*

Proof. Finding a team of experts from graph G while minimizing the sum of distances ($SD_G(T)$) is proved to be an NP-hard problem in [10]. Since $SD_G(T)$ is linearly related to $ComCost_G(T)$ (the objective function of Problem 1), then minimizing $ComCost_G(T)$ is also an NP-hard problem.

Since Problem 1 is an NP-hard problem, we have to rely on approximation or heuristic algorithms. Therefore, in the next section, we propose a 2-approximation algorithm and three heuristic algorithms for solving the problem.

4 Algorithms

4.1 Approximation Algorithm

In this section, we propose an approximation algorithm for solving Problem 1 with an approximation ratio of 2. The algorithm is based on converting the input graph (where the costs are associated with both nodes and edges) into a graph with costs/weights on only edges. The new graph G' has the same sets of nodes (experts) as the original graph G, but the node costs in G are moved onto edges in G'. In G', the edge weight between nodes u and v is defined as follows:

$$d'(u,v) = (1-\lambda)(t(u)+t(v)) \ + \ 2\lambda d(u,v) \tag{1}$$

where $t(x)$ is the cost of node/expert x in the original graph G, $d(u,v)$ is the shortest distance between experts u and v in G, and λ is the tradeoff between the communication and personnel costs.

Below we show that the *combined cost* of a team T of experts with respect to graph G is the same as the *Sum of Distances* of T with respect to the converted graph G'.

Lemma 1. *For any team T of experts from graph G for a project, the following holds:*

$$ComCost_G(T) = SD_{G'}(T)$$

where G' is converted from G by moving the node weights in G onto edges using Equation (1).

Proof. Let $T = \{\langle s_1, c_{s_1}\rangle, \langle s_2, c_{s_2}\rangle, \ldots, \langle s_p, c_{s_p}\rangle\}$. According to Definition 2,

$$SD_{G'}(T) = \sum_{i=1}^{p} \sum_{j=i+1}^{p} d'(c_{s_i}, c_{s_j})$$

where p is the number of required skills in the project. According to the definition of d', we have:

$$\begin{aligned} SD_{G'}(T) &= \sum_{i=1}^{p} \sum_{j=i+1}^{p} ((1-\lambda)(t(c_{s_i}) + t(c_{s_j})) + 2\lambda d(c_{s_i}, c_{s_j})) \\ &= (1-\lambda) \sum_{i=1}^{p} \sum_{j=i+1}^{p} (t(c_{s_i}) + t(c_{s_j})) + 2\lambda \sum_{i=1}^{p} \sum_{j=i+1}^{p} d(c_{s_i}, c_{s_j}) \\ &= (p-1)(1-\lambda) \sum_{i=1}^{p} t(c_{s_i}) + 2\lambda SD_G(T) \\ &= (p-1)(1-\lambda) PC(T) + 2\lambda SD_G(T) \\ &= ComCost_G(T) \end{aligned}$$

Based on the above lemma, finding a team of experts from graph G that minimizes the combined cost function (defined in Definition 4) is equivalent to finding a team of experts from graph G' that minimizes the *Sum of Distances* function (defined in Definition 2) based on d'. In [10], we proved that finding a team of experts while minimizing sum of distances is an NP-hard problem and proposed an approximation algorithm that finds a team of experts that minimizes the *Sum of Distances*. The approximation ratio of the algorithm is 2 as long as the pairwise distance function used in the *Sum of Distances* definition satisfies the triangle inequality. Below we show that function d' satisfies the triangle inequality.

Lemma 2. *The distance function $d'(u, v)$ defined in Equation (1) satisfies the triangle inequality.*

Proof. Function $d(u, v)$ in Equation (1) is the shortest distance between two nodes u and v in graph G. Since the shortest distance satisfies the triangle inequality, function d satisfies the triangle inequality. Thus, we have $d(a, b) \leq d(a, c) + d(c, b)$, where $\{a, b, c\}$ is an arbitrary triplet of nodes. Then, the following inequality holds since $0 \leq \lambda \leq 1$:

$$2\lambda d(a, b) \leq 2\lambda d(a, c) + 2\lambda d(c, b)$$

Thus,

$$\begin{aligned} (1-\lambda)(t(a) + t(b)) + 2\lambda d(a, b) &\leq (1-\lambda)(t(a) + t(b)) + 2\lambda d(a, c) + 2\lambda d(c, b) \\ &\leq 2(1-\lambda)t(c) + (1-\lambda)(t(a) + t(b)) + \\ &\quad 2\lambda d(a, c) + 2\lambda d(c, b) \end{aligned}$$

Based on the definition of d', the last inequality is equivalent to $d'(a, b) \leq d'(a, c) + d'(c, b)$. Since $\{a, b, c\}$ are chosen arbitrarily, d' satisfies the triangle inequality.

Since d' satisfies the triangle inequality, we can use the 2-approximation algorithm that we proposed in [10] to find a team T of experts in graph G' that minimizes the *Sum of Distances* function, $SD_{G'}(T)$. Based on Lemma 1, such a team also minimizes the combined cost, $ComCost_G(T)$, with respect to graph G. The pseudo-code of this algorithm is presented in Algorithm 1. The major difference between Algorithm 1 and the one in [10] is that Algorithm 1 takes two more inputs, namely, the expert costs t and the tradeoff λ between the communication and personnel costs, and uses distance function d' to compute the sum of distances of a team. The algorithm finds an approximate solution by using an expert ($e1$ in the pseudo-code) with a required skill as a seed in a team and adding its nearest expert with each of other required skills into the team. After checking all such teams, the team with the smallest sum of distances to the seed (calculated based on d') is returned as the best team. More explanation about the algorithm can be found in [10]. Note that two pre-built indexes are used as inputs to the algorithm. One is an inverted index for accessing $C(s_i)$, the set of experts in G having skill s_i. The other is a hash index that stores the shortest distances d of all pairs of experts in G. Both indexes can be pre-built because they are independent of the input project. The time complexity of this algorithm is $O(p^2 \times (C_{max})^2)$, where p is the number of required skills, and $C_{max} = \max_{1 \leq i \leq p} |C(s_i)|$ in which $|C(s_i)|$ is the cardinality of $C(s_i)$.

Theorem 2. *Algorithm 1 finds the team T of experts that minimizes $ComCost_G(T)$ with 2-approximation.*

Proof. In [10], we proved that Algorithm 1 is a 2-approximation algorithm for finding a team that minimizes the Sum of Distances. Since $SD_{G'}(T)$ based on d' is equivalent to $ComCost_G(T)$ according to Lemma 1, Algorithm 1 is a 2-approximation algorithm for finding a team of experts that minimizes the $ComCost_G(T)$.

4.2 Iterative Replace Algorithm

In this section, a heuristic algorithm is proposed for finding the best team of experts. The basic idea is as follows. Again, we use $C(s_i)$ to denote the set of experts that hold skill s_i. The algorithm consists of two phases. In the first phase, for each required skill s_i, the experts in $C(s_i)$ are sorted based on their cost in ascending order, and a team T is initialized by selecting the first expert in each $C(s_i)$ (i.e. the cheapest expert with the required skill s_i). This is the cheapest feasible team without consideration of the communication cost. In the second phase, each remaining expert in each $C(s_i)$ is tested to replace an expert in T which is currently assigned to skill s_i. If the replacement decreases the value of $ComCost_G(T)$, then the replace operation is applied permanently on T. The pseudo-code of the above procedure is presented in Algorithm 2. Note that $C(s_i)_j$ means the j-th expert in the sorted list of $C(s_i)$.

Since the algorithm needs to sort each $C(s_i)$, its run time is $O(C_{max} \log(C_{max}) + C_{max} \times p)$, where p is the number of required skills, and $C_{max} = \max_{1 \leq i \leq p} |C(s_i)|$. Note that if the experts in the initial team formed in the first phase are

Algorithm 1. The Approximation Algorithm for Finding Best Team

Input: project $P = \{s_1, s_2, \ldots, s_p\}$, set $C(s_i)$ of experts in graph G with skill s_i for $i = 1, \ldots, p$, distance d between each pair of nodes in G, cost t of each expert in G, and tradeoff λ between communication and personnel costs.

Output: the best team

```
$leastSumDistance \leftarrow \infty$
$bestTeam \leftarrow \emptyset$
for $i \leftarrow 1$ to $p$ do
    for each expert $e1 \in C(s_i)$ do
        $sumDistance \leftarrow 0$
        $T \leftarrow \{\langle s_i, e1 \rangle\}$
        for $j \leftarrow 1$ to $p$ and $j \neq i$ do
            $closestDistance = \infty$
            for each expert $e2 \in C(s_j)$ do
                $d'(e1, e2) \leftarrow (1 - \lambda)(t(e1) + t(e2)) + 2\lambda d(e1, e2)$
                if $d'(e1, e2) < closestDistance$ then
                    $closestDistance = d'(e1, e2)$
                    $closestNeighbor = e2$
            add $\langle s_j$, $closestNeighbor \rangle$ to $T$
            $sumDistance \leftarrow sumDistance + closestDistance$
        if $sumDistance < leastSumDistance$ then
            $leastSumDistance \leftarrow sumDistance$
            $bestTeam \leftarrow T$
return $bestTeam$
```

Algorithm 2. The Iterative Replace Algorithm for Finding Best Team

Input: project $P = \{s_1, s_2, \ldots, s_p\}$, set $C(s_i)$ of experts in graph G with skill s_i for $i = 1, \ldots, p$, distance d between each pair of nodes in G, cost t of each expert in G, and tradeoff λ between communication and personnel costs.

Output: the best team

```
for $i \leftarrow 1$ to $p$ do
    sort the experts in $C(s_i)$ based on their cost (i.e. $t$ function) in ascending order.
$T \leftarrow \emptyset$
for $i \leftarrow 1$ to $p$ do
    add $\langle s_i$, $C(s_i)_1 \rangle$ to the $T$
    $index_i \leftarrow 1$
while at least one $index_i$ has not reached $|C(s_i)|$ do
    for $i \leftarrow 1$ to $p$ do
        if $index_i < |C(s_i)|$ then
            $index_i \leftarrow index_i + 1$
            $T_{new} \leftarrow T$
            replace the holder of $s_i$ in $T_{new}$ with $C(s_i)_{index_i}$.
            if $ComCost_G(T_{new}) < ComCost_G(T)$ then
                $T \leftarrow T_{new}$
return $T$
```

disconnected from all the other experts with a required skill, no expert in the initial team will be replaced in the second phase, and thus the cheapest team is returned as the best team. However, the communication cost of the cheapest team can be very high (e.g., when any two members are disconnected from each other). Thus, when the graph is disconnected, this algorithm may return a poor team in terms of the combined cost. This problem can be alleviated by using the largest connected subgraph (i.e., the core of the graph) as the input data to the algorithm. Therefore, in our experiments, if the team produced by the algorithm is disconnected, we run the algorithm one more time on the core of the graph.

4.3 Minimal Cost Contribution Algorithm

In this section, another heuristic algorithm is proposed. The general structure of the algorithm is similar to Algorithm 1 in that they both seed a team with an expert with a required skill and add new members to the team to cover all the other required skills. The teams with different seeds are compared to select the best team. The difference is in how to expand the team with other experts. In Algorithm 1, the team was expanded by adding the nearest neighbor (according to the d' function) from each $C(s_j)$ where s_j is a required skill not covered by the seed expert. In that team expansion process, only the seed expert affects the addition of other members and the relationships among other experts in the team are not considered.

Algorithm 3. The Minimal Cost Contribution Algorithm for Finding Best Team Using Each of the Experts with a Required Skill as Initial Team Member

Input: project $P = \{s_1, s_2, \ldots, s_p\}$, set $C(s_i)$ of experts in graph G with skill s_i for $i = 1, \ldots, p$, distance d between each pair of nodes in G, cost t of each expert in G, and tradeoff λ between communication and personnel costs.

Output: the best team

```
$bestTeam \leftarrow$ NULL
$leastComCost \leftarrow \infty$
for $i \leftarrow 1$ to $p$ do
    for each expert $e_{initial} \in C(s_i)$ do
        $T \leftarrow \emptyset$
        add $\langle s_i, e_{initial} \rangle$ to $T$
        for $j \leftarrow 1$ to $p$ and $i \neq j$ do
            $leastMCC \leftarrow \infty$
            for each expert $e \in C(s_j)$ do
                Compute $MCC(e, T)$ using Formula (2)
                if $MCC(e, T) < leastMCC$ then
                    $leastMCC \leftarrow MCC(e, T)$
                    $expertWithLeastMCC \leftarrow e$
            add $\langle s_j, expertWithLeastMCC \rangle$ to $T$
        if $ComCost_G(T) < leastComCost$ then
            $leastComCost \leftarrow ComCost_G(T)$
            $bestTeam \leftarrow T$
return $bestTeam$
```

In the new algorithm proposed in this section, new members of a team are added incrementally and each new member is chosen by considering its communication costs with all the current members (not only the seed member) of the team in addition to the personnel cost of the new member. Assume that T is the current team of experts with some (but not all) of the required skills. To add into T an expert with an uncovered skill s_k, our new algorithm selects an expert e from the set $C(s_k)$ of experts with skill s_k that minimizes the following function:

$$MCC(e,T) = (1-\lambda)t(e) + \lambda\frac{\sum_{i=1}^{|T|} d(e,e_i)}{|T|} \tag{2}$$

where e_i is the expert responsible for the i-th skill in T and λ is the trade-off between communication and personnel costs. The first term of this function considers the personnel cost of expert e and the second considers the average communication cost between e and each of the current members of T. We refer to this function as Minimal Cost Contribution (MCC) function because the selected expert makes the minimal contribution to the total *ComCost* of the expanded new team compared to other experts being considered in $C(s_k)$.

The pseudo-code of the new algorithm, called the Minimal Cost Contribution (MCC) algorithm, is presented in Algorithm 3. The algorithm iterates through each expert with a required skill and uses the expert as the initial member of a candidate team T. For each candidate team T and each skill s_j uncovered by T, the algorithm incrementally selects and adds to T an expert e from $C(S_j)$ that minimizes the $MCC(e,T)$ function. The candidate team T is expanded in this way until all the required skills are covered by it. If the combined cost of T, $ComCost(T)$, is less than the least combined cost among all the previously generated candidates, T becomes the current best team (held in $bestTeam$). After all the candidate teams are generated (each with a different expert as the initial member), the team in $bestTeam$ is returned[1]. The time complexity of the MCC algorithm is $O(p^3 \times (C_{max})^2)$, where p is the number of required skills, and $C_{max} = \max_{1\leq i\leq p} |C(s_i)|$ in which $|C(s_i)|$ is the cardinality of $C(s_i)$.

[1] Note that the MCC algorithm is inspired by the Maximal Marginal Relevance (MMR) method [5] for increasing diversity in document retrieval results while maintaining high query relevance in the retrieved documents. The MMR method re-ranks the search results by using the most relevant document as the the first returned document and incrementally adding a new document to the result list by selecting a document that maximizes the MMR function which combines the relevance of the document and the dissimilarity of the document to the documents in the current result list. However, our MCC method is different from the MMR retrieval method in the following aspects. First, MMR generates the result with only one seed (i.e., the most relevant document) while MCC uses each expert with a required skill to initialize a candidate team and the final team is the one with the least *ComCost* among all the candidate teams. Second, the second term in the MMR function measures the maximal dissimilarity between the new document and the documents on the current result list, while MCC uses the average distance between the new team member and the current team members. We believe that the average distance better reflects the cost contribution of a new member to the team.

To improve the speed of the algorithm, we propose a variation of the *MCC* algorithm by using only an expert with the rarest required skill (i.e., the skill s whose $|C(s)|$ is the smallest among all the required skills) as the initial member of a candidate team. Thus, the number of candidate teams is reduced to the number of the skill holders of the the rarest required skill. We call this variation of the *MCC* algorithm *MCC-Rare*. Its time complexity is $O(p^2 \times C_{min} \times C_{max})$, where $C_{min} = \min_{1 \leq i \leq p} |C(s_i)|$.

5 Experimental Results

In this section, the proposed algorithms for finding a team of experts from a graph G which minimizes $ComCost_G(T)$ are evaluated. All the algorithms are implemented in Java. The experiments are conducted on an Intel(R) Core(TM) i7 2.80 GHz computer with 4 GB of RAM.

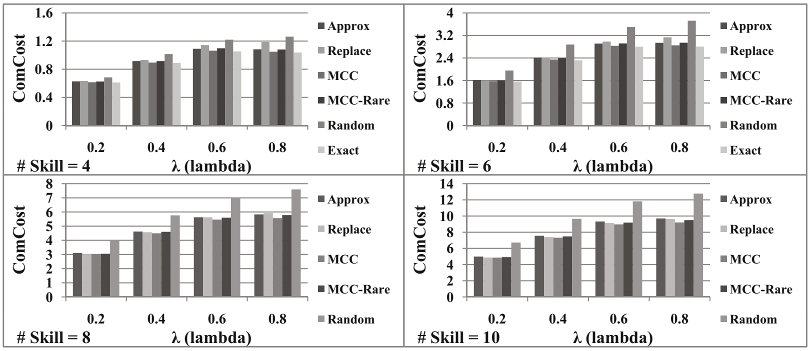


Fig. 1. The combined cost values of different methods on the DBLP dataset

5.1 The Datasets and Experimental Setup

To the best of our knowledge, real datasets that completely match our problem are not publicly available. Therefore, our proposed algorithms are evaluated on the datasets that were previously used in this domain [11,10]. The DBLP and IMDb data sets are used in the experiments. The DBLP XML data[2] is used for producing the DBLP graph. The dataset contains information about a set of papers and their authors. For each paper, the paper title, the author names, and the conference where it was published are specified. The same as in [11,10], only the papers of some major conferences in computer science are used for building the data graph, which include: SIGMOD, VLDB, ICDE, ICDT, EDBT, PODS, KDD, WWW , SDM, PKDD, ICDM, ICML, ECML, COLT, UAI, SODA, FOCS, STOC, and STACS. The set of experts and their skills are generated in the same way as in [11,10].

[2] http://dblp.uni-trier.de/xml/

The experts are the authors that have at least three papers in the DBLP. The skills of each expert is the set of keywords (terms) that appear in the titles of at least two papers of the expert. Two experts are connected together if they have at least two publications together. The weight of the edge between two experts n_i and n_j is equal to $1 - |\frac{p_{n_i} \cap p_{n_j}}{p_{n_i} \cup p_{n_j}}|$ where p_{n_i} is the set of papers of author n_i. The cost of an expert is set to be the number of publications of the expert. This is based on the assumption that the more publications an expert has, the more expertise the expert possesses, and thus the more expensive he/she is. The final graph has 5,658 nodes (experts) and 8,588 edges.

The same as [10], we use the part of the IMDb dataset which contains information about the actors and the list of movies that each actor played in[3]. It is preprocessed exactly the same as [10]. The communication cost between each pair of experts are also calculated the same as the DBLP dataset. Due to the space limit, more details are omitted. The cost of an expert is defined as the number of movies the actor plays in. The graph has 6,784 nodes and 35,875 edges. Due to the space limit, most of the results are only presented for the DBLP dataset. However, the results of the IMDb dataset show similar trend.

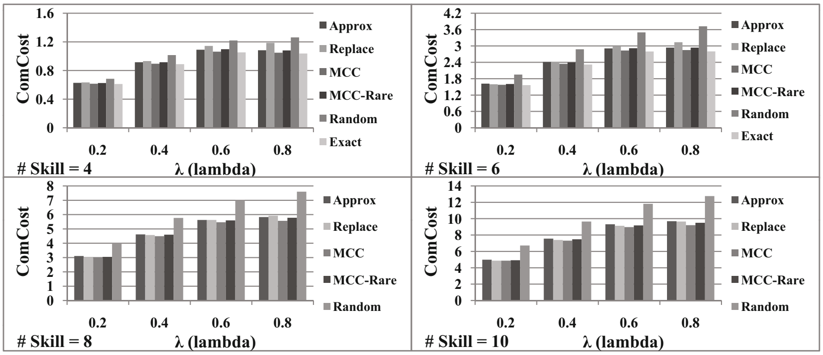
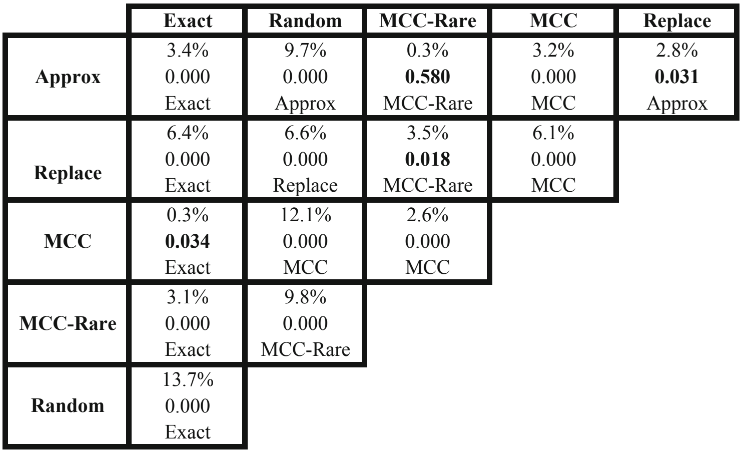


Fig. 2. The combined cost values of different methods on the IMDb dataset

The projects used in the experiments are generated as follows. We set the number of skills in a project to 4, 6, 8 or 10. For each number of skills, 50 sets of skills are generated randomly, corresponding to 50 random projects. The average result over the 50 projects for each number of skills is computed for each algorithm. As a baseline for the comparisons, we also include the results of a random method, which simply selects, among 10,000 random teams, the team with the lowest combined cost for the required set of skills. We also include the results of the *Exact* algorithm. It simply uses an exhaustive search to find the best answer among all possible teams.

[3] http://www.imdb.com/interfaces



	Exact	Random	MCC-Rare	MCC	Replace
Approx	3.4% 0.000 Exact	9.7% 0.000 Approx	0.3% **0.580** MCC-Rare	3.2% 0.000 MCC	2.8% **0.031** Approx
Replace	6.4% 0.000 Exact	6.6% 0.000 Replace	3.5% **0.018** MCC-Rare	6.1% 0.000 MCC	
MCC	0.3% **0.034** Exact	12.1% 0.000 MCC	2.6% 0.000 MCC		
MCC-Rare	3.1% 0.000 Exact	9.8% 0.000 MCC-Rare			
Random	13.7% 0.000 Exact				

Fig. 3. Results of t-test to show the level of difference between the algorithms on the DBLP dataset. The number of required skills is set to 4 and λ is set to 0.5.

5.2 Evaluation on Combined Cost

Figures 1 and 2 show the average combined cost values of teams for different algorithms for DBLP and IMDb datasets receptively. Please note that the results of the *Exact* algorithm is only provided for four and six skills. It is because by increasing the number of required skills, the number of possible teams grows exponentially. Thus, the *Exact* algorithm does not terminate in reasonable time for eight or more skills. The results show that all of the algorithms outperform the *Random* method. The results also suggest that the *MCC* method has the lowest cost values among non-exact methods. The results of *MCC-Rare* and *Approx* are very similar. In most of the cases, the *Replace* method has higher combined cost value than other proposed methods. *MCC-Rare* uses only an expert with the rarest required skill as the initial member of a candidate team. Therefore, *MCC-Rare* considers less number of candidate teams than *MCC* and thus its performance in terms of the combined cost is worse than the *MCC* algorithm.

To see whether the results of different algorithms are significantly different from each other, we run a t-test on each pair of methods. The results are shown in Figure 3. In each cell, the first number shows the percentage difference between the two methods (i.e., the absolute difference between the two values divided by the average of the two values). The second number is the p value from the t-test and the third row indicates which method has lower combined cost value (e.g. the *Exact* method always has the lowest combined cost.). In terms of percentage difference, the closest method to the *Exact* method is *MCC*. Their percentage difference is only 0.3%. Also, the p-values indicate that all pairs of methods are significantly different from each other except for *MCC-Rare* and *Approx*, which are not significantly different although *MCC-Rare* is slightly better.

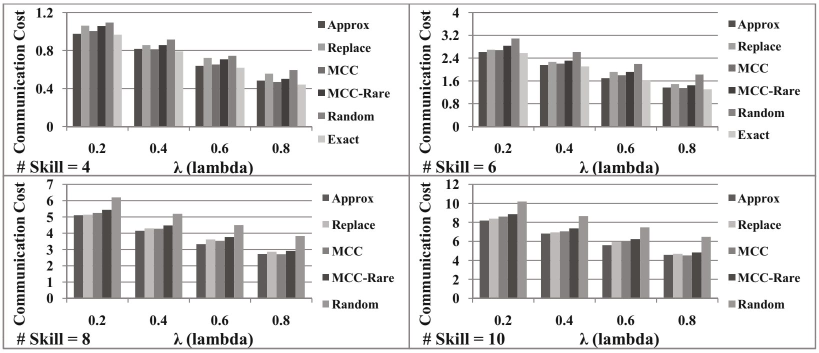


Fig. 4. The communication cost values of different methods on the DBLP dataset

5.3 Evaluation on Communication Cost

Figure 4 shows the average communication cost values of teams for different algorithms. The same as previous section, the results of the *Exact* algorithm is only provided for four and six skills. Please note that none of the algorithms explicitly minimize the communication cost. However, all of them implicitly minimize it by minimizing the combined cost function. The results suggest that the *Approx* algorithm has the lowest communication cost than among non-exact methods. In addition, for four skills, its results are very close to the one for the *Exact* method. Please note that by increasing the value of λ, the communication cost decreases. This is an expected result based on Definition 4.

5.4 Evaluation on Personnel Cost

Figure 5 shows the average personnel cost values of teams for different algorithms. As can be seen, *MCC-Rare* has the lowest personnel cost on all skill

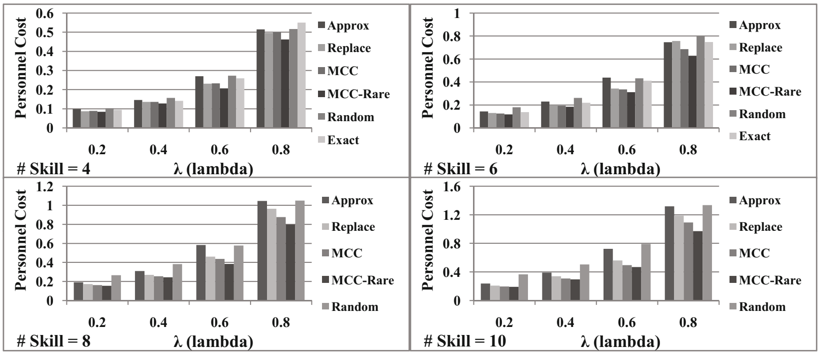


Fig. 5. The personnel cost values of different methods on the DBLP dataset

Table 1. Run time in milliseconds of different algorithms on DBLP. λ is set to 0.5.

# Req. Skills	Approx	Replace	MCC	MCC-Rare	Random	Exact
3	2	1	3	1	23	1,331
4	3	1	9	2	37	4,537
5	7	1	15	3	46	86,115
6	28	2	63	5	98	1,415,856

numbers and for all λ values, even lower than the exact algorithm. Note that the exact algorithm always finds the team with the lowest combined cost, which may not have the lowest personnel cost. The results also show that the *Random* method has the highest personnel cost,and the *Approx* algorithm has the second highest personnel cost. Please note that by increasing the value of λ, the personnel cost increases. This is also an expected result based on Definition 4.

5.5 The Run Time

Table 1 provides the run time of each method. It shows that the *Replace* algorithm is the fastest among others. *MCC-Rare* and *Approx* are the second and third respectively. *MCC* is the slowest among the 4 proposed methods, but still much faster than the *Random* method. The results also show the unreasonable run time of the *Exact* method and its inapplicability in practice.

6 Conclusions

We have proposed four algorithms for finding a team of experts in a social network that minimizes both the communication cost and the personnel cost of the team. The first algorithm is an approximation algorithm with a provable performance bound. The other three algorithms use heuristics to find sub-optimal solutions. Our experiments show that the *MCC* method has the lowest combined cost among the non-exact methods, but its run time is higher than other proposed heuristic or approximation algorithms. *MCC-Rare* reduces the run time of *MCC* and has the second lowest combined cost. The *Approx* algorithm has similar combined cost to *MCC-Rare* with a bit higher run time. The *Replace* method is the fastest but with the highest combined cost among the proposed methods. All the proposed methods are much faster than the *Random* and *Exact* methods. The *Random* method has the highest cost. The results indicate that the proposed methods are both effective and efficient.

References

1. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi., S.: Power in unity: Forming teams in large-scale community systems. In: Proc. of CIKM 2010 (2010)

2. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Online team formation in social networks. In: Proc. of the WWW 2012 (2012)
3. Backstrom, L., Huttenlocher, D., Kleinberg, J., Lan, X.: Group formation in large social networks: membership, growth, and evolution. In: Proc. of KDD 2006 (2006)
4. Baykasoglu, A., Dereli, T., Das, S.: Project team selection using fuzzy optimization approach. Cybern. Syst. 38(2), 155–185 (2007)
5. Carbonell, J., Goldstein, J.: The use of mmr, diversity-based reranking for reordering documents and producing summaries. In: Proc. of SIGIR 1998 (1998)
6. Dorn, C., Dustdar, S.: Composing near-optimal expert teams: A trade-off between skills and connectivity. In: Proc. of the International Conference on Cooperative Information Systems (2010)
7. Fitzpatrick, E.L., Askin, R.G.: Forming effective worker teams with multi functional skill requirements. Comput. Ind. Eng. 48(3), 593–608 (2005)
8. Gaston, M., Simmons, J., des Jardins, M.: Adapting network structures for efficient team formation. In: Proc. of the AAAI Fall Symposium on Artificial Multi-agent Learning (2004)
9. Jackson, M.: Network formation. The New Palgrave Dictionary of Economics and the Law (2008)
10. Kargar, M., An, A.: Discovering top-k teams of experts with/without a leader in social networks. In: Proc. of CIKM 2011 (2011)
11. Lappas, T., Liu, L., Terzi, E.: Finding a team of experts in social networks. In: Proc. of KDD 2009 (2009)
12. Li, C., Shan, M.: Team formation for generalized tasks in expertise social networks. In: Proc. of IEEE International Conference on Social Computing (2010)
13. Wi, H., Oh, S., Mun, J., Jung, M.: A team formation model based on knowledge and collaboration. Expert Syst. Appl. 36(5), 9121–9134 (2009)
14. Zakarian, A., Kusiak, A.: Forming teams: an analytical approach. IIE Transactions 31, 85–97 (2004)

Feature-Enhanced Probabilistic Models for Diffusion Network Inference

Liaoruo Wang*, Stefano Ermon*, and John E. Hopcroft

Department of Computer Science
Cornell University, Ithaca, NY 14853, USA
{lwang,ermonste,jeh}@cs.cornell.edu

Abstract. Cascading processes, such as disease contagion, viral marketing, and information diffusion, are a pervasive phenomenon in many types of networks. The problem of devising intervention strategies to facilitate or inhibit such processes has recently received considerable attention. However, a major challenge is that the underlying network is often unknown. In this paper, we revisit the problem of inferring latent network structure given observations from a diffusion process, such as the spread of trending topics in social media. We define a family of novel probabilistic models that can explain recurrent cascading behavior, and take into account not only the time differences between events but also a richer set of additional features. We show that MAP inference is tractable and can therefore scale to very large real-world networks. Further, we demonstrate the effectiveness of our approach by inferring the underlying network structure of a subset of the popular Twitter following network by analyzing the topics of a large number of messages posted by users over a 10-month period. Experimental results show that our models accurately recover the links of the Twitter network, and significantly improve the performance over previous models based entirely on time.

1 Introduction

Cascading processes, such as the spread of a computer virus or an infectious disease, are a pervasive phenomenon in many networks. Diffusion and propagation processes have been studied in a broad range of disciplines, such as information diffusion [1–4], social networks [5, 6], viral marketing [7, 8], epidemiology [9], and ecology [10]. In previous work, researchers have mostly focused on a number of optimization problems derived from cascading processes, where the goal is to devise intervention strategies to either maximize (e.g., viral marketing) or minimize (e.g., network interdiction, vaccination programs) the propagation. However, these studies often assume that the underlying network is known to the observer, which in practice is not true in many situations.

In this paper, we revisit the problem of inferring latent network structure given observations of a diffusion process. For example, by observing a disease epidemic, we want to infer the underlying social contact network, or by observing the spread of trending topics, we want to estimate the connectivity of the social media. Fig. 1 illustrates a case of information diffusion in the popular Twitter network. The nodes represent a subset

* Corresponding authors.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 499–514, 2012.

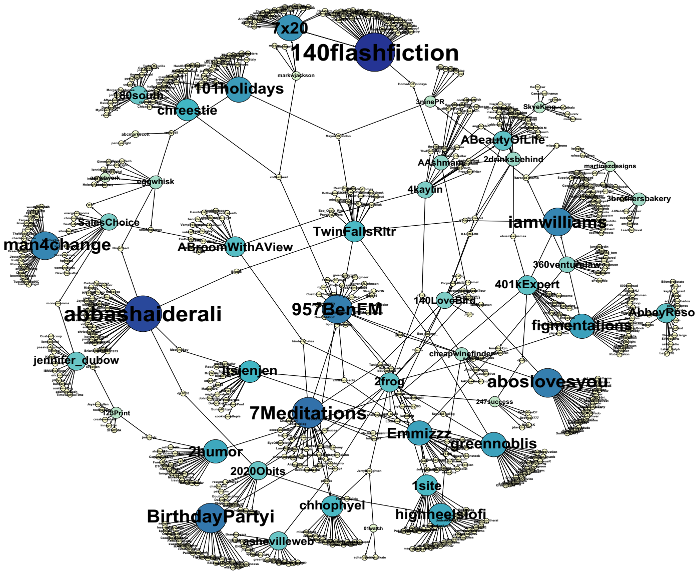


Fig. 1. Information diffusion in the Twitter network (see PDF for colored version)

of the Twitter users that have posted about a common trending topic, and the directed edges represent the "following" relation between the users. There is a clear pattern in the figure. The bigger and darker nodes, followed by the smaller and lighter nodes, form the hubs of the diffusion process. By looking at the time-stamps of the messages and at the underlying network structure, we observe that most information flows initiate at a hub node and spread across the network to reach other hub nodes and their followers. However, it is non-trivial to come up with such a picture simply by looking at the time-stamps of the messages, since without knowing the underlying network structure, we cannot decide from whom a node copied the information from. Intuitively, messages carry implicit information about the social relations among users. For instance, users who repeatedly post messages about the same topic within a short period of time, are more likely to be connected. Thus, a motivating application of this paper is to what extent we can estimate the relations in social networks by analyzing the messages published by users over time.

This type of latent network inference problem based on the time-stamps of infection (or, information-reproduction) events has received increasing interest over the past few years [1–3]. Previous work was largely based on two major assumptions: 1) the diffusion process is causal (i.e., not affected by events in the future), and 2) infection events closer in time are more likely to be causally related (e.g., according to an exponential, Rayleigh, or power-law distribution). While the causality assumption is indeed crucial and always satisfied in practice, we realize that there are many other factors that can be highly informative as far as the causality relations are concerned. For example, the time-stamps at which two users publish their tweets are important to decide whether they are related, but other factors such as the language or the content of the messages can be as important. Even if the two messages are close in time, they are unlikely to be related if the messages are written in different languages. Further, previous models in the literature are mostly focused on monotonic processes, while real-world processes are often recurrent. For instance, it is very common for one user to post about the same topic multiple times on Twitter, or purchase the same item regularly on Amazon.

Contributions. Motivated by these challenges, we define a family of novel probabilistic models that generalize previous models based solely on time. We propose a primary approach MONET that can handle recurrent diffusion processes. Further, we consider a richer set of additional features for infection events, defining novel feature-enhanced models that can better explain the observed data. With distributed optimization and convex objective functions, we can efficiently solve the problem of inferring the most probable latent network structure. Using additional features such as the languages of the messages and Jaccard indexes between the messages, we can accurately recover the links of the Twitter network by analyzing the topics of a large number of messages posted by a subset of the Twitter users over a 10-month period. Experimental results show that our models significantly improve the accuracy of the estimates over previous models by as much as 78.7%.

2 Problem Definition

We consider a diffusion process across a network represented by a directed, weighted graph $G = (V, E)$. Let $\mathbf{A} = \{\alpha_{jk} | j, k \in V, j \neq k\}$ be the adjacency matrix of weights. A directed edge (j, k) has weight $\alpha_{jk} \geqslant 0$ that denotes the pairwise transmission rate from node j to node k. For example, in the case of an infectious disease spreading through a population, V represents a group of individuals and E represents the strength of the social contacts among them. In the case of an invasive species colonizing a new territory, V represents patches of land and E represents the connectivity between them. We assume that the diffusion process is stochastic but *causal*, that is, it depends on the past history but not on the future. Specifically, we consider a diffusion process that starts with one or more nodes, and spreads across the network subject to an independent local probabilistic model of "infection", where a node infects its neighbors independently of the status of other nodes in the network [5].

When studying such diffusion processes, the underlying network is often unknown (latent). However, we assume that one can observe a set of *cascades* of "infection" (or, information-reproduction) events. A *cascade* is a sequence of infection events

$$\pi = \{(v_0, t_0), \cdots, (v_N, t_N)\}$$

during a given time interval T, where $v_i \in V$ is a node that becomes infected at time t_i. T is the *horizon* of cascade π. Note that different cascades may have different horizons. For example, in the Twitter network, each cascade corresponds to a trending topic, and we have an entry (v_i, t_i) for each tweet posted by user v_i at time t_i. Given a probabilistic model $P(\pi|G)$ that gives the probability of observing a certain cascade π when the underlying network is G, the problem of inferring the latent network structure from

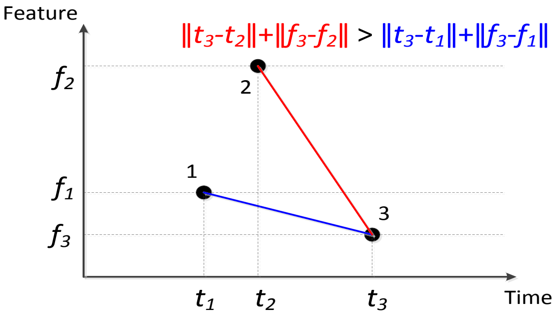


Fig. 2. Feature-enhanced probabilistic model

observed cascades has received considerable attention [1–3]. It is usually assumed that V is known but E is unknown, and the goal is to find a (weighted) graph

$$G^* = \arg\max_G \prod_{c=1}^{M} P(\pi^c|G)$$

that maximizes the probability of observing cascades $\pi^1, \cdots, \pi^M$, which are assumed to be i.i.d. (independent, identically distributed) realizations of the underlying diffusion process.

The building block to define $P(\cdot)$ is the likelihood function $f(t_k|t_j; \alpha_{jk})$ that gives the probability density that node v_j infected at time t_j infects node v_k at time t_k (see below for more details). Such models are centered on the time differences between the infection events of a cascade, and exploit the causal nature of the diffusion process by setting the likelihood to zero whenever $t_k - t_j < 0$. Further, they assume that events closer in time are more likely to be causally related. For instance, if node v_k became infected shortly after node v_j was infected, this is considered as an indication that the two events are causally related (i.e., v_k was infected by v_j). These time-based models are the foundation of our work.

While time is indeed a crucial element of the network inference problem, in practical applications, observations of a diffusion process often carry additional key information. For example, the diagnosis of an infection often comes with additional information about the specific strain. When a topic or a rumor spreads through a social network, one can also observe the context in which it appears. This motivates our definition of a *generalized cascade*

$$\pi_g = \{(v_0, t_0, \mathbf{f}_0), \cdots, (v_N, t_N, \mathbf{f}_N)\} \tag{1}$$

where $v_i \in V$ is a node infected at time t_i, and $\mathbf{f}_i \in \mathcal{F}$ is a feature vector describing the additional information available for the i-th infection event. Using the additional information contained in a generalized cascade, we can define a generalized feature-enhanced probabilistic model where the probability of a transmission event depends not only on the time differences, but also on the additional features. Specifically, we use a probability density function $f(t_k, \mathbf{f}_k|t_j, \mathbf{f}_j; \alpha_{jk})$ as a building block, which depends causally on the relative time difference $t_k - t_j$ as well as on the additional features $\mathbf{f}_k$ and $\mathbf{f}_j$. Fig. 2 gives an example of the difference between previous models that are based solely on time and a case of our feature-enhanced models where $f(t_k, \mathbf{f}_k|t_j, \mathbf{f}_j; \alpha_{jk})$ depends on $||t_k - t_j|| + ||\mathbf{f}_k - \mathbf{f}_j||$. Node 3 is considered to be more related to node 1 than node 2 by our feature-enhanced models, while it is determined to be more related to node 2 by models based only on time.

Furthermore, previous models are focused on monotonic diffusion processes, while most real-world processes are recurrent. For example, it is common for one user to post about the same topic multiple times on Twitter, or purchase the same item multiple times on Amazon. Repeated posts of the same topic show a higher level of interest in that topic, and exchanged posts of the same topic between a group of nodes also show a higher level of connectivity in that group. We take these factors into account in our feature-enhanced models and assign respective reward/penalty to each scenario.

There are two different ways of modeling a diffusion process where nodes can be infected multiple times in one cascade. The first model considers an infection event as the result of all previous events, and thus we call it *non-splitting*. By contrast, the second model considers an infection event of a node as the result of all previous events *up to its last infection*. This model is memoryless and thus we call it *splitting*. We mainly focus on the non-splitting model in this section, but the results can be extended to the splitting case. We will later present experimental results in Section 4 for both non-splitting and splitting models.

2.1 Generalized Cascade Model

We first recall some standard notation from previous literature, and then define our feature-enhanced models based on generalized cascades.

Recap. Recall the standard notation from [1] and [11]. Given that node j was infected at time t_j, the *survival function* of edge (j, k) is the probability that, by time t_k, node k was not infected by node j. That is,

$$S\left(t_k|t_j; \alpha_{jk}\right) = 1 - F\left(t_k|t_j; \alpha_{jk}\right), \tag{2}$$

where α_{jk} denotes the transmission rate from node j to node k, and $F\left(t_k|t_j; \alpha_{jk}\right)$ is the cumulative distribution function. Further, the *hazard function* (or, instantaneous infection rate) of edge (j, k) is given by

$$H\left(t_k|t_j; \alpha_{jk}\right) = \frac{f\left(t_k|t_j; \alpha_{jk}\right)}{S\left(t_k|t_j; \alpha_{jk}\right)}, \tag{3}$$

where

$$f\left(t_k|t_j; \alpha_{jk}\right) = \left.\frac{d}{dt} F\left(t|t_j; \alpha_{jk}\right)\right|_{t_k}$$

is the likelihood function. Table 1 shows the survival and hazard functions based on the exponential, Rayleigh, and power-law distribution.

Table 1. Parametric Models [1]

Model	Likelihood Function $f\left(t_k \| t_j;\alpha_{jk}\right)$	Survival Function $S\left(t_k \| t_j;\alpha_{jk}\right)$	Hazard Function $H\left(t_k \| t_j;\alpha_{jk}\right)$
Exponential	$\begin{cases}\alpha_{jk}e^{-\alpha_{jk}\left(t_k-t_j\right)} & \text{if } t_j<t_k\\ 0 & \text{otherwise}\end{cases}$	$e^{-\alpha_{jk}\left(t_k-t_j\right)}$	α_{jk}
Rayleigh	$\begin{cases}\alpha_{jk}\left(t_k-t_j\right)e^{-\alpha_{jk}\frac{\left(t_k-t_j\right)^2}{2}} & \text{if } t_j<t_k\\ 0 & \text{otherwise}\end{cases}$	$e^{-\alpha_{jk}\frac{\left(t_k-t_j\right)^2}{2}}$	$\alpha_{jk}\left(t_k-t_j\right)$
Power Law	$\begin{cases}\frac{\alpha_{jk}}{\delta}\left(\frac{t_k-t_j}{\delta}\right)^{-1-\alpha_{jk}} & \text{if } t_j<t_k-\delta\\ 0 & \text{otherwise}\end{cases}$	$\left(\frac{t_k-t_j}{\delta}\right)^{-\alpha_{jk}}$	$\frac{\alpha_{jk}}{t_k-t_j}$

Multiple Occurrences. Real-world diffusion processes are often recurrent, that is, we often observe multiple occurrences of the same node in one cascade. With multiple occurrences of node k and node j, the survival function for the non-splitting case is

$$\begin{aligned} S\left(t_k|t_j;\alpha_{jk}\right) &= \prod_{k:t_k^{(1)}\leqslant T^c}\ \prod_{1\leqslant i\leqslant N_k^c}\ \prod_{j\neq k:t_j^{(1)}<t_k^{(i)}} S\left(t_k^{(i)}|t_j;\alpha_{jk}\right)\\ &= \prod_{k:t_k^{(1)}\leqslant T^c}\ \prod_{1\leqslant i\leqslant N_k^c}\ \prod_{j\neq k:t_j^{(1)}<t_k^{(i)}}\ \prod_{1\leqslant \ell\leqslant N_j^c(t_k^{(i)})} S\left(t_k^{(i)}|t_j^{(\ell)};\alpha_{jk}\right), \end{aligned}$$

where T^c is the horizon of cascade π^c, and $t_k^{(i)}, i\in\{0,\cdots,N_k^c,N_k^c+1\}$ denote the time-stamps of node k infections in cascade π^c. We assign two special time-stamps for every node: $t_k^{(0)}=0$ and $t_k^{(N_k^c+1)}=T^c$. N_k^c denotes the number of node k infections in cascade π^c. $N_j^c(t_k^{(i)})$ denotes the number of node j infections before the i-th infection of node k. Similarly, the hazard function is given by

$$H\left(t_k^{(i)}|t_j^{(\ell)};\alpha_{j,k}\right)=\frac{f\left(t_k^{(i)}|t_j^{(\ell)};\alpha_{j,k}\right)}{S\left(t_k^{(i)}|t_j^{(\ell)};\alpha_{j,k}\right)}.$$

Additional Features. Consider two feature vectors $\mathbf{f}_k,\mathbf{f}_j\in\mathcal{F}$ associated with node $k,j\in V$ in a cascade. Let $d\left(\mathbf{f}_k,\mathbf{f}_j\right)$ denote the distance between the two feature vectors. We include an extra term $e^{-d(\mathbf{f}_k,\mathbf{f}_j)}$ in the likelihood function to reflect this distance factor. For example, given an exponential distribution, we have

$$f\left(t_k,\mathbf{f}_k|t_j,\mathbf{f}_j;\alpha_{jk}\right)=\begin{cases}\gamma\alpha_{jk}e^{-d\left(\mathbf{f}_k,\mathbf{f}_j\right)}e^{-\alpha_{jk}\left(t_k-t_j\right)}, & \text{if } t_j<t_k\\ 0, & \text{otherwise}\end{cases}$$

where γ is a normalization constant. Thus, the survival function is given by

$$\begin{aligned} S\left(t_k|t_j,\mathbf{f}_j;\alpha_{jk}\right) &= 1-F\left(t_k|t_j,\mathbf{f}_j;\alpha_{jk}\right)=1-\int_{\mathcal{F}}\int_{t_j}^{t_k} f\left(t,\mathbf{f}|t_j,\mathbf{f}_j;\alpha_{jk}\right)\,\mathrm{d}t\,\mathrm{d}\mathbf{f}\\ &= e^{-\alpha_{jk}\left(t_k-t_j\right)}\int_{\mathcal{F}}\gamma e^{-d\left(\mathbf{f},\mathbf{f}_j\right)}\mathrm{d}\mathbf{f}=e^{-\alpha_{jk}\left(t_k-t_j\right)}. \end{aligned} \tag{4}$$

Then, the hazard function is

$$H\left(t_k, \mathbf{f}_k | t_j, \mathbf{f}_j; \alpha_{jk}\right) = \frac{f\left(t_k, \mathbf{f}_k | t_j, \mathbf{f}_j; \alpha_{jk}\right)}{S\left(t_k | t_j, \mathbf{f}_j; \alpha_{jk}\right)} = \begin{cases} \gamma \alpha_{jk} e^{-d\left(\mathbf{f}_k, \mathbf{f}_j\right)}, & \text{if } t_j < t_k \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

2.2 MAP Inference

Given independent cascades, the likelihood of a set of cascades $\{\pi_g^1, \cdots, \pi_g^M\}$ is the product of the likelihood of each cascade:

$$\prod_{1 \leqslant c \leqslant M} f\left(\pi_g^c; \mathbf{A}\right), \tag{6}$$

where $\mathbf{A} = \{\alpha_{jk} | j, k \in V, j \neq k\}$ is a weighted adjacency matrix of transmission rates. Given a cascade π_g^c, the probability that node k was not infected by time T^c is the product of the survival functions of the infected nodes. The formulation can be extended according to the generalized cascade model discussed in Section 2.1. For example, if a node was infected multiple times during the observation window, this repeated lack of ability to infect node k should also be considered. Specifically, the probability that node k was not infected by time T^c is

$$\prod_{j: t_j^{(1)} \leqslant T^c} \prod_{1 \leqslant i \leqslant N_j^c} S\left(T^c | t_j^{(i)}, \mathbf{f}_j^{(i)}; \alpha_{jk}\right).$$

Given the parents of the infected nodes, infections are assumed to be conditionally independent. Thus, the likelihood of the observed cascade π_g^c is

$$f\left(\pi_g^c; \mathbf{A}\right) = \prod_{j: t_j^{(1)} \leqslant T^c} \prod_{1 \leqslant i \leqslant N_j^c} f\left(t_j^{(i)}, \mathbf{f}_j^{(i)} | \pi_g^c \backslash \left(j, t_j^{(i)}, \mathbf{f}_j^{(i)}\right); \mathbf{A}\right).$$

Given the i-th infection of node k, the likelihood of node j being its first parent is

$$\begin{aligned} f\left(t_k^{(i)}, \mathbf{f}_k^{(i)} | t_j, \mathbf{f}_j; \mathbf{A}\right) = & \prod_{s \neq j: t_s^{(1)} < t_k^{(i)}} \prod_{1 \leqslant p \leqslant N_s^c\left(t_k^{(i)}\right)} S\left(t_k^{(i)} | t_s^{(p)}, \mathbf{f}_s^{(p)}; \alpha_{s,k}\right) \\ & \times \sum_{1 \leqslant \ell \leqslant N_j^c\left(t_k^{(i)}\right)} f\left(t_k^{(i)}, \mathbf{f}_k^{(i)} | t_j^{(\ell)}, \mathbf{f}_j^{(\ell)}; \alpha_{j,k}\right) \prod_{q \neq \ell} S\left(t_k^{(i)} | t_j^{(q)}, \mathbf{f}_j^{(q)}; \alpha_{j,k}\right). \end{aligned}$$

Thus, the likelihood of the observed cascade π_g^c is

$$f\left(\pi_g^c; \mathbf{A}\right) = \prod_{k: t_k^{(1)} \leqslant T^c} \prod_{1 \leqslant i \leqslant N_k^c} \left(\sum_{j: t_j^{(1)} < t_k^{(i)}} f\left(t_k^{(i)}, \mathbf{f}_k^{(i)} | t_j, \mathbf{f}_j; \mathbf{A}\right) \right).$$

Combine the two equations above and include the condition $s = j$, we have

$$\begin{aligned} f\left(\pi_g^c; \mathbf{A}\right) = \prod_{k: t_k^{(1)} \leqslant T^c} \prod_{1 \leqslant i \leqslant N_k^c} & \left(\prod_{s: t_s^{(1)} < t_k^{(i)}} S\left(t_k^{(i)} | t_s, \mathbf{f}_s; \alpha_{s,k}\right) \times \right. \\ & \left. \sum_{j: t_j^{(1)} < t_k^{(i)}} \sum_{1 \leqslant \ell \leqslant N_j^c\left(t_k^{(i)}\right)} \frac{f\left(t_k^{(i)}, \mathbf{f}_k^{(i)} | t_j^{(\ell)}, \mathbf{f}_j^{(\ell)}; \alpha_{j,k}\right)}{S\left(t_k^{(i)} | t_j^{(\ell)}, \mathbf{f}_j^{(\ell)}; \alpha_{j,k}\right)} \right). \end{aligned}$$

Add the information that some nodes were never infected during the horizon T^c, and then,

$$f\left(\pi_g^c;\mathbf{A}\right)=\prod_{k:t_k^{(1)}\leqslant T^c}\prod_{1\leqslant i\leqslant N_k^c}\prod_{t_m>T^c}S\left(T^c|t_k^{(i)},\mathbf{f}_k^{(i)};\alpha_{i,m}\right)\times$$
$$\prod_{s:t_s^{(1)}<t_k^{(i)}}S\left(t_k^{(i)}|t_s,\mathbf{f}_s;\alpha_{s,k}\right)\times\sum_{j:t_j^{(1)}<t_k^{(i)}}\sum_{1\leqslant\ell\leqslant N_j^c(t_k^{(i)})}H\left(t_k^{(i)},\mathbf{f}_k^{(i)}|t_j^{(\ell)},\mathbf{f}_j^{(\ell)};\alpha_{j,k}\right).\tag{7}$$

Eq. (7) gives the likelihood of cascade π_g^c for the non-splitting model. However, for the splitting case, this likelihood is given by

$$\begin{aligned}f\left(\pi_g^c;\mathbf{A}\right)=&\prod_{k:t_k^{(1)}\leqslant T^c}\prod_{1\leqslant i\leqslant N_k^c}\prod_{t_m>T^c}S\left(T^c|t_k^{(i)},\mathbf{f}_k^{(i)};\alpha_{i,m}\right)\times\\&\prod_{s:t_s^{(1)}<t_k^{(i)}}\prod_{N_s^c(t_k^{(i-1)})<p\leqslant N_s^c(t_k^{(i)})}S\left(t_k^{(i)}|t_s^{(p)},\mathbf{f}_s^{(p)};\alpha_{s,k}\right)\times\\&\sum_{j:t_j^{(1)}<t_k^{(i)}}\sum_{N_j^c(t_k^{(i-1)})\leqslant\ell\leqslant N_j^c(t_k^{(i)})}H\left(t_k^{(i)},\mathbf{f}_k^{(i)}|t_j^{(\ell)},\mathbf{f}_j^{(\ell)};\alpha_{j,k}\right).\end{aligned}\tag{8}$$

Eq. (8) is similar to Eq. (7) except that we only consider the segment between the $(i-1)$-th and i-th occurrence of node k for the survival and hazard function.

Problem Definition. Our goal is to infer the connectivity and estimate the infection rate α_{jk} for each pair of nodes (j,k) such that the likelihood of observed cascades $\{\pi_g^1,\cdots,\pi_g^M\}$ is maximized. Specifically,

$$\begin{aligned}&\text{minimize}_{\mathbf{A}} && -\sum_{1\leqslant c\leqslant M}\log f\left(\pi_g^c;\mathbf{A}\right)\\&\text{subject to} && \alpha_{jk}\geqslant 0,\ j,k\in V,j\neq k.\end{aligned}\tag{9}$$

where $\mathbf{A}=\{\alpha_{jk}|j,k\in V,j\neq k\}$ are the variables. The inferred edges of the network are those pairs of nodes with infection rate $\alpha_{jk}>0$.

3 Proposed Approach: MONET

In this section, we discuss the properties of the optimization problem arising from the MAP inference task in our feature-enhanced probabilistic models defined in Section 2. By Eq. (6) and Eq. (7), the log-likelihood of cascades $\{\pi_g^1,\cdots,\pi_g^M\}$ is

$$L\left(\{\pi_g^1,\cdots,\pi_g^M\};\mathbf{A}\right)=\sum_{1\leqslant c\leqslant M}\Phi_1(\pi_g^c;\mathbf{A})+\Phi_2(\pi_g^c;\mathbf{A})+\Phi_3(\pi_g^c;\mathbf{A}),\tag{10}$$

where for each cascade $\pi_g^c\in\{\pi_g^1,\cdots,\pi_g^M\}$,

$$\begin{aligned}
\Phi_1(\pi_g^c; A) &= \sum_{k:t_k^{(1)} \leqslant T^c} \sum_{1 \leqslant i \leqslant N_k^c} \sum_{s:t_s^{(1)} < t_k^{(i)}} \log S\left(t_k^{(i)} | t_s, \mathbf{f}_s; \alpha_{s,k}\right), \\
\Phi_2(\pi_g^c; A) &= \sum_{k:t_k^{(1)} \leqslant T^c} \sum_{1 \leqslant i \leqslant N_k^c} \log \sum_{j:t_j^{(1)} < t_k^{(i)}} \sum_{1 \leqslant \ell \leqslant N_j^c(t_k^{(i)})} H\left(t_k^{(i)}, \mathbf{f}_k^{(i)} | t_j^{(\ell)}, \mathbf{f}_j^{(\ell)}; \alpha_{j,k}\right), \\
\Phi_3(\pi_g^c; A) &= \sum_{k:t_k^{(1)} \leqslant T^c} \sum_{1 \leqslant i \leqslant N_k^c} \sum_{t_m^{(1)} > T^c} \log S\left(T^c | t_k^{(i)}, \mathbf{f}_k^{(i)}; \alpha_{k,m}\right).
\end{aligned}$$

The above equations are a strict generalization of the ones presented in [1], that is, we recover the same formulation where there are no multiple occurrences and we do not consider any additional features other than time. Further, we also generalize several results of the model in [1] to our feature-enriched setting, for both splitting and non-splitting cases. Formally,

Theorem 1. *The following results hold:*

- *Given any distance functions, log-concave survival functions, and concave hazard functions, the problem defined by Eq.* (9) *is convex in* **A**.
- *The optimization problem defined by Eq.* (9) *is convex for the feature-enhanced models with exponential, Rayleigh, or power law distribution.*
- *The solution to Eq.* (9) *gives a consistent maximum likelihood estimator.*

The proof of Theorem 1 is similar to that of [1], and is omitted for space reasons.

We call our primary approach MONET, which provides non-splitting and splitting solutions for the network inference problem defined by Eq. (9) where nodes can be repeatedly infected.

Analyzing MONET. We discuss some properties of the solution to the optimization problem defined in Eq. (9) for the generalized feature-enhanced models with the exponential, Rayleigh, and power-law distribution. This is equivalent to maximizing the log-likelihood defined in Eq. (10). Clearly, the expression in Eq. (10) depends on the transmission rate α_{jk} and the relative time difference $t_k - t_j$ between each occurrence of node j and node k. Note that it does not depend on the absolute values of the time-stamps. In general, however, Eq. (10) depends on the absolute values of the feature vectors (i.e., it depends not only on the distance between observed feature vectors), due to the normalization constant γ.

As discussed in [1], Φ_1 and Φ_3 encourage sparse solutions by imposing negative weights on **A**. Specifically, Φ_1 penalizes α_{jk} based on the relative time difference $t_k - t_j$ and Φ_3 penalizes α_{ki} for uninfected node i based on $T^c - t_k$ (i.e., until the horizon cut-off). Note that MONET only infers impossible edges based on 0 transmission rates. Due to finite observation window, the lack of ability to infect some node i within time T^c does not mean it is impossible to infect node i (i.e., there is no edge).

The term Φ_2 emphasizes the intuition that infected nodes must have at least one parent (appearing before them in a cascade) by which they were infected. If this is not ensured, $\Phi_2 = -\infty$ will be negatively unbounded. The additional features used in our models only affect the term Φ_2. Specifically, infected nodes tend to select those that are more similar to them as their parents. Further, in the non-splitting case, only the first occurrence of each node in the cascade is affected by this hard constraint (further occurrences can still be explained by the parent of the first occurrence). However, simply using the first explanation can be too penalizing, and adding more parents might improve the likelihood.

Computational Aspects. As in previous work, we can parallelize the solution to the optimization problem defined in Eq. (9). Given a network with n nodes, this optimization problem has $O(n^2)$ variables, but the objective function can be separated into n independent sub-problems with $O(n)$ variables each. For each node $k = 1, \cdots, n$, we optimize the k-th column of the matrix $\mathbf{A}$ of transmission rates, solving for $(n-1)$ unknown transmission rates $\{\alpha_{jk}\}$ where $j \neq k$. To compute the k-th column, we only require the infection times of the nodes in those cascades where node k appears. Optimal columns are joined to form a globally optimal transmission rate matrix.

If node j never appears before node k in any cascade, we have no evidence to suggest the existence of a directed edge (j, k). That is, α_{jk} only contributes to the non-positive term Φ_2 in Eq. (10). Thus, in every iteration, we set α_{jk} to the optimal value 0 to simplify the objective function $L\left(\{\pi_g^1, \cdots, \pi_g^M\}; \mathbf{A}\right)$.

Any convex optimization package can be used to solve the optimization problem. However, regular packages such as CVXOPT [12] could not handle the scale of our Twitter dataset and ran out of memory. Thus, we use the limited-memory BFGS algorithm with box constraints (L-BFGS-B) [13] to solve Eq. (9) and Eq. (10) by implicitly approximating the inverse Hessian matrix. We use the box constraints to enforce the non-negativity of the transmission rates.

4 Experimental Results

We evaluate the performance of our models by analyzing the diffusion of information in the popular Twitter network. Using a dataset crawled from January to October 2010 that contains 9,409,063 tweets published by 66,679 Twitter users, we analyze the cascading behavior of some trending topics and try to infer the underlying network structure.

4.1 Experimental Setup

Dataset Description. We conduct experiments on a subset of the Twitter network, which contains 66,679 nodes and 240,637 directed links. Each node represents a Twitter user and each edge represents a following relation. Contrary to previous work [1, 2], the adjacency matrix (ground truth) of this subgraph has also been crawled and thus is entirely known. In order to identify trending topics, we group the messages posted by these users according to their Hashtags[1]. We assume that messages containing the same Hashtag form a (generalized) cascade of a particular topic. Note that certain cascades corresponding to popular Hashtags might not be explained by our generative models. For example, #iphone is a widespread Hashtag that users often proactively include in their tweets rather than passively copy from another user. Therefore, we select a subset

[1] Hashtags are words or phrases prefixed with the symbol # to label groups and topics.

of not-so-popular Hashtags (e.g., #Mokpo and #GagaSouthAmerica2011), which are more specific and "local". That is, we consider those "local" cascades such that if node k writes about a Hashtag at time t_k, then it must have followed (or, have copied from) some node that wrote about the same Hashtag before time t_k.

This assumption is particularly important to our experiments. Since we only have a subset of the whole Twitter network, infected nodes observed in a cascade might have copied the information from some node that does not belong to this subset of users. Since we observe that MONET performs better on the cascades where the "locality" intuition holds, we trace the propagation of 500 Hashtags (that consist of 103,148 tweets) across the Twitter network from January to October 2010[2]. The average length of the cascades is 166, and there are a total of 2,521 unique users. On average, over 75% of the users post multiple times of the same Hashtag in each cascade. The inference is focused on the top 200 users that belong to the largest number of cascades. The size of the dataset is such that this inference problem can be solved by NETRATE and NETINF.

Feature Model. When collecting the Hashtags, we also record the entire message (or, tweet) containing the Hashtag. This represents the additional feature $\mathbf{f}_j$ for each node $j \in V$ in the generalized cascade model. In this paper, we use two primary distance metrics associated with texts: language and Jaccard index.

◇ **Language.** We observe that messages belonging to the same cascade (i.e., with the same Hashtag) are often written in several languages. For example, a cascade starting with an English tweet can spread to multilingual users who post tweets in Italian or Chinese but keep the original Hashtag. Intuitively, tweets in different languages, even if published closely in time, should not be considered as an implication of connectivity. Let $\ell(\cdot)$ be a function mapping a tweet to its language. We define a distance function with respect to language

$$d_L(\mathbf{f}_i, \mathbf{f}_j) = \begin{cases} 0, & \ell(\mathbf{f}_i) = \ell(\mathbf{f}_j); \\ 1, & \ell(\mathbf{f}_i) \neq \ell(\mathbf{f}_j). \end{cases}$$

The language information is computed using the n-gram model proposed in [14]. Note that this language identification algorithm provides noisy estimates.

◇ **Pairwise similarity.** We include pairwise similarity (a.k.a. Jaccard index) as another distance metric in our models. Given two tweets $\mathbf{f}_j$ and $\mathbf{f}_k$ posted by node j and node k, the distance function with respect to Jaccard index is defined as

$$d_J(\mathbf{f}_i, \mathbf{f}_j) = 1 - J_{jk} = 1 - \frac{|\mathbf{f}_j \cap \mathbf{f}_k|}{|\mathbf{f}_j \cup \mathbf{f}_k|},$$

where we consider the tweets as sets of words. Intuitively, besides the time factor, node k is more likely to have copied the information from node j if their tweets have higher similarity.

◇ **Combination.** We also consider both language and Jaccard similarity as a combined feature, defining another distance function

$$d_{L+J}(\mathbf{f}_i, \mathbf{f}_j) = w_J d_J(\mathbf{f}_i, \mathbf{f}_j) + w_L d_L(\mathbf{f}_i, \mathbf{f}_j).$$

[2] We will release an anonymized dataset due to Twitter's data privacy policy.

where w_J and w_L are the weights associated with the language and the Jaccard similarity feature. Since Jaccard similarity is typically small, we use a weight w_J to ensure that its contribution is comparable to that of the language distance.

Note that the normalization constant γ in Eq. (5) is hard to compute, since it involves a summation over all possible messages of up to 140 characters (the maximum length allowed by Twitter). In our experiments, we consider γ as fixed and independent of $\mathbf{f}_j$. In the case of language, this is equivalent to assuming that there are roughly the same number of possible messages for any given language.

Our optimization framework contains a hierarchical set of models for the MAP inference problem: splitting/non-splitting with multiple occurrences (MONET), language (MONET+L), Jaccard index (MONET+J), and their combination (MONET+LJ).

Evaluation Measures. To evaluate the performance of our feature-enhanced models, we consider the following aspects:

- ◇ **Baseline.** We use NETRATE [1] and NETINF [2] as two baselines to compare with our models. Since repeated occurrences are not allowed in NETRATE, we keep exactly one copy of each node and remove all other duplicates from each cascade. We use the true number of edges as an input parameter for NETINF. Due to license issues with the optimization software, we do not compare with CONNIE [3] in this paper, but its performance is comparable with that of NETRATE and NETINF according to previous literature.
- ◇ **Quantitative performance.** We use precision, recall, and F1-score to evaluate the performance of our models against the baselines. These measures focus on the number of correct pairs of nodes inferred. For example, given a pair of nodes (k, j) such that k is following j, if our method suggests that $\alpha_{jk} > 0$ (i.e., information flows from j to k), then consider it as a true positive (TP). False positives (FP) and false negatives (FN) are defined in a similar way.
- ◇ **Efficiency.** We evaluate the efficiency (i.e., elapsed time required for obtaining the optimum) of our feature-enhanced models.

All algorithms are implemented using Python with the Fortran implementation of L-BFGS-B available in Scipy [15], and all experiments are performed on a machine running CentOS Linux with a 6-core Intel x5690 3.46GHZ CPU and 48GB memory.

4.2 Quantitative Performance

We trace the propagation of a set of 500 Hashtags that consist of 103,148 tweets across a subset of the Twitter network that contains 66,679 nodes and 240,637 directed links. We want to infer the connectivity of the top 200 users that appear in the largest number of these 500 cascades. We evaluate our models against NETRATE and NETINF by comparing the inferred network and the ground truth via three metrics: precision, recall, and F1-score. F1-score, the harmonic mean of precision and recall, measures the accuracy of the estimates. The primary model MONET handles the basic scenario where nodes can have multiple occurrences in one cascade. As discussed in Section 2.1, MONET can be extended to consider a set of additional features, such as language (MONET+L), Jaccard similarity (MONET+J), and both (MONET+LJ).

Table 2. Performance comparison on Twitter (non-splitting/exponential)

METRIC				METHOD		
	NetInf	NetRate	moNet	moNet+L	moNet+J	moNet+LJ
Precision	0.362	0.592	0.434	0.464	0.524	0.533
Recall	0.362	0.069	0.307	0.374	0.450	0.483
F1-score	**0.362**	**0.124**	**0.359**	**0.414**	**0.484**	**0.507**
TP	518	99	439	535	644	692
FP	914	62	573	618	586	606
FN	914	1333	993	897	788	740

Table 3. Performance comparison on Twitter (splitting/exponential)

METRIC				METHOD		
	NetInf	NetRate	moNet	moNet+L	moNet+J	moNet+LJ
Precision	0.362	0.592	0.514	0.516	0.531	0.534
Recall	0.362	0.069	0.599	0.605	0.618	0.635
F1-score	**0.362**	**0.124**	**0.554**	**0.557**	**0.571**	**0.581**
TP	518	99	858	867	885	910
FP	914	62	810	812	781	793
FN	914	1333	574	565	547	522

Upper Bound of Recall. Similar to previous models, moNet requires node j to appear at least once before node k for $\alpha_{jk} > 0$ to be possibly inferred (i.e., information flows from j to k). For our dataset, no more than 86.4% of the edges in the ground truth can be recovered given the cascades. This represents an upper bound on recall for any probabilistic model based on the causality of the diffusion process.

Exponential Model. Table 2 and Table 3 compare the precision, recall, and F1-score of our non-splitting and splitting models introduced in Section 2.2 with NetRate and NetInf according to the exponential distribution (see Table 1). NetRate tends to be highly conservative when estimating the connectivity of the Twitter network, and thus has good precision but very low recall. NetInf knows how many edges there are in the true network, and slightly improves over NetRate. Without knowing the ground truth, moNet balances the precision-recall trade-off and improves the accuracy over NetRate by 65.5% for the non-splitting case and 77.6% for the splitting case. As expected, moNet+L, moNet+J, and moNet+LJ further improve the F1-score on top of moNet with the help of additional features. In particular, moNet+LJ improves the accuracy by as much as 78.7% over NetRate and 37.7% over NetInf for the splitting case.

Rayleigh Model. Table 4 and Table 5 compare the precision, recall, and F1-score of our non-splitting and splitting models with NetRate and NetInf according to the Rayleigh distribution (see Table 1). Similarly, without knowing the ground truth, moNet balances the precision-recall trade-off and improves the accuracy over NetRate by 55.7%

Table 4. Performance comparison on Twitter (non-splitting/Rayleigh)

Metric	Method					
	NetInf	NetRate	moNet	moNet+L	moNet+J	moNet+LJ
Precision	0.354	0.560	0.420	0.454	0.479	0.484
Recall	0.354	0.072	0.218	0.262	0.286	0.294
F1-score	**0.354**	**0.127**	**0.287**	**0.332**	**0.358**	**0.366**
TP	507	103	312	375	409	421
FP	925	81	430	451	445	449
FN	925	1329	1120	1057	1023	1011

Table 5. Performance comparison on Twitter (splitting/Rayleigh)

Metric	Method					
	NetInf	NetRate	moNet	moNet+L	moNet+J	moNet+LJ
Precision	0.354	0.560	0.480	0.493	0.495	0.499
Recall	0.354	0.072	0.562	0.566	0.570	0.572
F1-score	**0.354**	**0.127**	**0.518**	**0.527**	**0.530**	**0.533**
TP	507	103	805	811	816	819
FP	925	81	872	835	834	821
FN	925	1329	627	621	616	613

for the non-splitting case and 75.5% for the splitting case. moNet+L, moNet+J, and moNet+LJ further improve the F1-score on top of moNet with the help of additional features. In particular, moNet+LJ improves the accuracy by as much as 76.2% over NetRate and 33.4% over NetInf for the splitting case.

Remarks. We have similar observations for the performance comparison according to the power-law distribution, but the tables are omitted here due to the space limitation. Our results suggest that the splitting model performs better than the non-splitting one, with much more true positives and far fewer false negatives. This suggests that the information diffusion in the Twitter network is better approximated by a memoryless process. Specifically, how a message posted by a Twitter user will be retweeted is not relevant to that user's previous history. Further, the exponential model provides slightly more accurate estimates over the Rayleigh one. The performance improvement achieved using the language information is smaller compared to that achieved using Jaccard similarity, but moNet+L improves over moNet and moNet+LJ improves over moNet+J. This suggests that the language feature does provide some useful information, although its effectiveness is likely to be limited by the noisy estimates provided by the language detection algorithm we use in our experiments.

4.3 Efficiency

Solving each of the sub-problems defined in the basic model moNet (i.e., optimizing one column of the transmission rate matrix $\mathbf{A}$) takes about 2 minutes on average using

L-BFGS-B with the history parameter $m = 10$. The running time, however, depends on the specific column being optimized, and ranges from a few seconds to several minutes. Introducing additional features requires an additional preprocessing time (in the order of minutes) to precompute the languages of the messages and the Jaccard indexes between the messages, but it does not significantly affect the running time of the optimization procedure.

5 Related Work

A substantial amount of work has been devoted to the task of studying cascading processes in large-scale networks. Largely motivated by marketing applications, the predominant focus over the past decade has been on optimization problems, where the goal is to maximize the spread of a certain cascade through a given network, either by selecting a good subset of nodes to initiate the cascade [5] or by applying a broader set of intervention strategies such as node and edge additions [7, 10]. As networks and networked systems are playing an increasingly important role in a number of disciplines, ranging from the interconnections between financial systems to epidemiology and ecology, researchers have recently begun to consider the problem of inferring the unknown (latent) underlying network given some observed cascading behavior [1–3]. Specifically, several generative probabilistic models have been developed to explain cascading behaviors, where the task of inferring the underlying network is tractable, involving the optimization of submodular [2] or convex objective functions [1, 3]. These models have been shown to perform well on a number of synthetic datasets, but there has been very limited experimentation on real-world scenarios. Moreover, the MemeTracker dataset [2] commonly used in previous work has no ground truth.

There are several obstacles when trying to apply these models to real-world problems, such as inferring the latent structure of a social network based on the diffusion of trending topics. Specifically, cascades are often formed by a mixed set of sub-cascades and it is difficult to obtain i.i.d. samples. However, real-world cascades also present a range of new opportunities to define richer probabilistic models. Previous work combined latent features with explicit ones to solve structural link prediction problems [16]. In this paper, we propose a feature-enhanced framework to address the scenario where nodes can be repeatedly infected. We develop a family of novel probabilistic models based not only on the time intervals between infection events, but also on a set of additional features, such as the content and the language of the messages exchanged in social media.

6 Conclusions

In this paper, we propose a family of feature-enhanced probabilistic models to infer the latent network structure from observations of a diffusion process. We develop a primary model called MONET with non-splitting and splitting solutions that can explain recurrent processes where nodes can be repeatedly infected (i.e., multiple occurrences in one cascade). Further, our models take into account not only the time differences between infection events, but also a richer set of features. The MAP inference problem, which still involves the optimization of a convex objective function, can be decomposed into smaller sub-problems that we can efficiently solve in parallel. Our experiments on the Twitter network show that our models successfully recover the underlying network structure, and significantly improve the performance over previous models based solely on time.

Acknowledgement. This research was partially supported by the U.S. AFOSR Grant FA9550-09-1-0675 and NSF Grant 0832782.

References

1. Gomez-Rodriguez, M., Balduzzi, D., Schölkopf, B.: Uncovering the temporal dynamics of diffusion networks. In: ICML, pp. 561–568 (2011)
2. Gomez-Rodriguez, M., Leskovec, J., Krause, A.: Inferring networks of diffusion and influence. In: KDD, pp. 1019–1028 (2010)
3. Myers, S.A., Leskovec, J.: On the convexity of latent social network inference. In: NIPS (2010)
4. Adar, E., Adamic, L.A.: Tracking information epidemics in blogspace. In: Web Intelligence, pp. 207–214 (2005)
5. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD (2003)
6. Lappas, T., Terzi, E., Gunopulos, D., Mannila, H.: Finding effectors in social networks. In: KDD (2010)
7. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: KDD (2002)
8. Watts, D.J., Dodds, P.S.: Influentials, networks, and public opinion formation. Journal of Consumer Research 34(4) (2007)
9. Wallinga, J., Teunis, P.: Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. American Journal of Epidemiology 160(6) (2004)
10. Sheldon, D., Dilkina, B., Elmachtoub, A., Finseth, R., Sabharwal, A., Conrad, J., Gomes, C., Shmoys, D., Allen, W., Amundsen, O., et al.: Maximizing the spread of cascades using network design. In: UAI (2010)
11. Lawless, J.F.: Statistical models and methods for lifetime data (1982)
12. Dahl, J., Vandenberghe, L.: CVXOPT: A Python package for convex optimization. In: Proc. Eur. Conf. Op. Res. (2006)
13. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math. Softw. 23(4), 550–560 (1997)
14. Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: Proc. 3rd Annual Symposium on Document Analysis and Information Retrieval (1994)
15. Jones, E., Oliphant, T., Peterson, P.: Scipy: Open source scientific tools for Python (2001), http://www.scipy.org/
16. Menon, A.K., Elkan, C.: Link Prediction via Matrix Factorization. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 437–452. Springer, Heidelberg (2011)

Influence Spread in Large-Scale Social Networks – A Belief Propagation Approach

Huy Nguyen and Rong Zheng

University of Houston, Computer Science Department
4800 Calhoun Rd., Houston, TX, 77204
{hanguyen4,rzheng}@uh.edu

Abstract. Influence maximization is the problem of finding a small set of seed nodes in a social network that maximizes the spread of influence under a certain diffusion model. The Greedy algorithm for influence maximization first proposed by Kempe, later improved by Leskovec suffers from two sources of computational deficiency: 1) the need to evaluate many candidate nodes before selecting a new seed in each round, and 2) the calculation of the influence spread of any seed set relies on Monte-Carlo simulations. In this work, we tackle both problems by devising efficient algorithms to compute influence spread and determine the best candidate for seed selection. The fundamental insight behind the proposed algorithms is the linkage between influence spread determination and belief propagation on a directed acyclic graph (DAG). Experiments using real-world social network graphs with scales ranging from thousands to millions of edges demonstrate the superior performance of the proposed algorithms with moderate computation costs.

1 Introduction

The social network of interactions among a group of individuals plays a fundamental role in the spread of information, ideas, and influence. Such effects have been observed in real life, when an idea or an action gains sudden widespread popularity through *"word-of-mouth"* or *"viral marketing"* effects. For example, free e-mail services such as Microsoft's Hotmail, later Google's Gmail, and most recently Google's Google+ achieved wide usage largely through referrals, rather than direct advertising. Another more recent example is the Hewlett-Packard (HP) TouchPad fire sale event [1]. The company slashed the price of TouchPad by 75% to clear out inventory. Without any mass media advertisement or public announcement, the move inadvertently generated an Internet phenomenon – with Twitter and Facebook users sharing tips on websites where the product was still in stock – and long lines at retailers as consumers jostled to pick up TouchPads.

In viral marketing, one important question is given limited advertisement resources, which set of customers should be targeted such that the resulting influenced population is maximized. Consider a social network modeled as a graph with vertices representing individuals and edges representing connections or relationship between two individuals. Under a specific diffusion model, the goal

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 515–530, 2012.

of influence maximization (IM) is to find k vertices (seed nodes) in the graph such that the expected number of vertices influenced by the k seeds is maximized [2,3,4]. Kempe *et al.* proved the submodularity of the influence spread function and suggested a greedy scheme (henceforth referred to as Greedy algorithm) with an incremental oracle that identifies, in each iteration, a new seed that maximizes the incremental spread. The approach was proven to be a $(1 - 1/e)$-approximation of the IM problem. However, there are major limitations with this method as previously mentioned. Follow-up works either only addresses one of the deficiencies [5,6] or sacrifices accuracy for less computation time [7].

In this work, we first establish the linkage between influence spread computation and belief propagation on a Bayesian network (modeled as a directed acyclic graph – DAG), where the marginal conditional dependency corresponds to the influence probabilities. Belief propagation has been extensively studied in literatures, and thus many exact or approximation algorithms can be leveraged to estimate the influence spread. For a general graph that contains loops, we propose two approximation algorithms that prune some edges in the graph to obtain a DAG that captures the bulk of influence spread. To reduce the number of candidate seed nodes, we localize the influence spread region such that at each round, only nodes that are affected by the previous selected seed need to be evaluated. Experimental study shows that the proposed algorithms can scale up to massive graphs with millions of edges with high accuracy. On real-world social network graphs, the proposed algorithms can achieve influence spread comparable to that by Greedy algorithm and incur significant less computation costs. They also outperform the scheme in [8] in achievable influence spread at the expense of marginal increase in computation time.

The main contributions of this paper are summarized as follows:

- We cast the problem of inference spread computation on a DAG as an instance of belief propagation on a Bayesian Network.
- We prove the #P-hardness of inference spread computation on a DAG.
- Two heuristics are proposed to construct DAGs from a general graph that capture the bulk of influence spread.
- A fast algorithm is devised to incrementally update the DAG as more seeds are added, and select candidate seeds.

The rest of this paper is organized as follows. In Section 2, we give a comprehensive review of the existing literature on influence spread maximization. Section 3 presents theoretical results concerning influence spread on DAGs. In Section 4, we devise two heuristics to reduce a general directed graph into a DAG which captures the majority of influence spread. Improvements on seed selection are discussed in Section 5. In Section 6, extensive experiment results are presented. Finally we conclude the paper and discuss future research directions in Section 7.

2 Related Work

In an effort to improve Greedy, Leskovec *et al.* [5] recognized that not all remaining nodes need to be evaluated in each round and proposed the "Cost-Effective Lazy Forward" (CELF) scheme. Experimental results demonstrate that CELF optimization could achieve as much as 700-time speed-up in selecting seeds. However, even with the CELF, the number of candidate seeds is still large. Recently, Goyal *et al.* proposed CELF++ [6] that has been shown to run from 35% to 55% faster than CELF. However, the improvement comes at the cost of higher space complexity to maintain a larger data structure to store the look-ahead marginal gains of each node.

Chen *et al.* devises several heuristic algorithms for influence spread computation [7,8,9]. In Degree Discount [7], the expected number of additional vertices influenced by adding a node v in the seed set is estimated based on v's one hop neighborhoods. It also assumes that the influence probability is identical on all edges. In [8] and [9], two approximation algorithms, PMIA and LDAG are proposed to compute the maximum influence set under IC and LT models, respectively. In LDAG, it has been proven that under the LT model, computing influence spread in a DAG has linear time complexity, and a heuristic on local DAG construction is provided to further reduce the compute time. We have proven in Section 3 that computing influence spread in a DAG under the IC model remains #P-hard. The marked difference between the two results arises from the fact that in the LT model, the activation of incoming edges is coupled so that in each instance, only one neighbor can influence the node of interest in an equivalent random graph model.

Another line of work explores diffusion models beyond LT and IC. Even-Dar *et al.* [10] argue that the most natural model to represent diffusion of opinions in a social network is the probabilistic voter model where in each round, each person changes his opinion by choosing one of his neighbors at random and adopting the neighbor's opinion. Interestingly, they show that the straightforward greedy solution, which picks the nodes in the network with the highest degree, is optimal. Sylvester [11] studies the spread maximization problem on dynamic networks and examines the use of dynamic measures with Greedy algorithm on both LT and IC models. Chen *et al.* [12] consider a new model that incorporates negativity bias and design an algorithm to compute influence on tree structures.

3 Influence Spread on Directed Acyclic Graphs

In this section, we consider the problem of computing influence spread given a fixed seed set when the underlying social network is a DAG. We first show the problem remains #P-hard, and then establish its equivalence to the computation of marginal probabilities in a Bayesian network.

3.1 Problem Formulation

We consider a directed graph $\mathcal{G} = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges. For every edge $(u, v) \in E$, $p(u, v)$ denotes the probability of influence being propagated on the edge. In this paper, we adopt the Independent Cascade (IC) model. Given a seed set $S \subseteq V$, the IC model works as follows. Let $S_t \subseteq V$ be the set of node (newly) activated at time t, with $S_0 = S$ and $S_t \cap S_{t-1} = \emptyset$. At round $t+1$, every node $u \in S_t$ tries to activate its neighbors in $v \in V \setminus \bigcup_{0 \leq i \leq t} S_i$ independently with probability $p(u,v)$. The influence spread of S, denoted by $\sigma(S)$, is the *expected* number of activated nodes given seed set S.

Kempe *et. al* [4] proved two important properties of the $\sigma(\cdot)$ function: 1) $\sigma(\cdot)$ is *submodular*, namely, $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(T \cup \{v\}) - \sigma(T)$ for all $v \in V$ and all subsets S and T with $S \subseteq T \subseteq V$; 2) $\sigma(S)$ is *monotone*, i.e. $\sigma(S) \leq \sigma(T)$ for all set $S \leq T$. For any given spread function $\sigma(\cdot)$ that is both submodular and monotone, the problem of finding a set S of size k that maximizes $\sigma(S)$ can be approximated by a simple greedy approach.

3.2 Hardness of Computing Influence Spread on DAGs

In [4], Kempe *et. al* proposed an equivalent process of influence spread under the IC model, where at the initial stage, an edge (u,v) in $\mathcal{G}$ is declared to be *live* with probability $p(u,v)$ resulting in a subgraph of $\mathcal{G}$. A node u is active if and only if there is at least one path from some node in S to u consisting entirely of *live edges*. In general graphs, the influencer-influencee relationship may differ in one realization to another for bi-directed edges. In a DAG, on the other hand, such relationship is fixed and is independent of the outcome of the coin flips at the initial stage (other than the fact that some of the edges may not be present). Let $x_u, u \in V$ denotes the binary random variable of the active state of node u, namely, $\mathbb{P}(x_u = 1) = p(u)$. For each node v in S, $\mathbb{P}(x_v = 1) = 1$. If a node $u \notin S$ does not have any parent in $\mathcal{G}$ then $\mathbb{P}(x_u = 1) = 0$. From $\mathcal{G}$, the conditional probability $p(x_u | x_{Par(u)})$ is uniquely determined by the edge probability, where $x_{Par(u)}$ denotes the states of the parents of node u. In other words, influence spread can be modeled as a Bayesian network. If node u does not have any parent, $p(x_u | x_{Par(x_u)}) = p(x_u)$. The joint distribution is thus given by,

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_{Par(x_i)}). \tag{1}$$

Given the outcome of coin flips C, $\sigma_C(S) = \sum_{u \in V} x_u$. Therefore,

$$\sigma(S) = \mathbb{E}(\sigma_C(S)) = \sum_{u \in V} \mathbb{E}(x_u) = \sum_{u \in V} p(u). \tag{2}$$

The second equality is due to the linearity of expectations. To compute $p(u)$, we can sum (1) over all possible configurations for $x_v, v \in V \backslash u$. Clearly, such a naive approach has complexity that is exponential in the network's treewidth. In fact, the marginalization problem is known to be #P-complete on a DAG. However, since computing influence spread on a DAG can be reduced to a special instance of the marginalization problem, it remains to be shown if the former problem is #P-complete. The main result is summarized in the following theorem[1].

Theorem 1. *Computing the influence spread $\sigma(S)$ on a DAG given a seed set S is #P-complete.*

3.3 Estimating $\sigma(\cdot)$ via Belief Propagation

Belief propagation is a message passing algorithm for performing inference on graphical models, such as Bayesian networks and Markov random fields. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes [13]. For *singly-connected* DAGs, where between any two vertices there is only one simple path, the belief propagation (BP) algorithm [14] computes the exact solution with $O(n)$ complexity. For multi-connected DAGs, where multiple simple paths may exist between two vertices, belief propagation and many of its variants [13,15,16] have been shown to work well in general. Exact solutions such as junction tree [15] may incur the worst case complexity exponential to the number of vertices due to the need to enumerate all cliques in the DAG.

BP algorithms take as input a factor graph or a Bayesian Network. For each factor in the graph or a Bayesian node, a conditional probability table (CPT) is constructed. For a node v with the parent set $Par(v) = \{par_1, par_2, \ldots, par_k\}$, its CPT consists of one column for each state and one row for each set of states its parents may assume. In influence spread, each state has two states: active (1) and inactive (0). Thus the number of rows in a CPT is 2^k. An illustrative example of a factor graph and one of its CPT's is given in Figure 1 and 2.

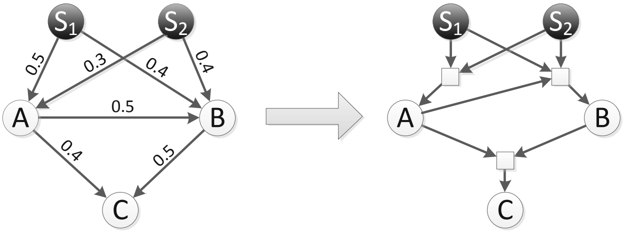


Fig. 1. Converting a DAG into a factor graph

Once the factor graph and CPT's associated with each factor are available, we can apply a suitable BP algorithm to calculate the active probability of each node in the DAG. $\sigma(\cdot)$ can then be determined by (2).

[1] All proofs are omitted due to lack of space but can be found on the full technical report at http://arxiv.org/abs/1204.4491

		States of C	
A	B	0	1
0	0	1	0
0	1	0.5	0.5
1	0	0.6	0.4
1	1	0.3	0.7

Fig. 2. CPT of C with two parents A, B

Computation Complexity. The complexity of $\sigma(\cdot)$ calculation is dominated by the execution of the BP algorithm. A variety of BP algorithms exist. In our evaluation, we adopt the Loopy Belief Propagation (LBP) algorithm which was shown to perform well for various problems [17,18]. LBP's complexity to estimate the active probability of a node v is $O(M^d)$, where M is the number of possible labels (states) for each variable ($M = 2$), and d is the number of neighbors of v.

3.4 A Single Pass Belief Propagation Heuristic for $\sigma(\cdot)$ Estimation

Calculating $\sigma(\cdot)$ with LBP produces highly accurate results, but the computation time remains to be high when the graph is multi-connected. The main complexity arises from the fact that the activation of parents of a node may be correlated in a multi-connected graph. Thus, in computing the activation probability of the node, one needs to account for the joint distribution of its parent nodes. Next, we propose a single pass belief propagation (SPBP) algorithm that ignores such correlation in determining $\sigma(\cdot)$. Note that the heuristic is exact when the graph is singly-connected. Let $\mathcal{D}(\cdot)$ be the input DAG. Consider a node $v \in \mathcal{D}(\cdot)$. Given the activation probabilities of its parents $Par(v)$, we approximate $p(v)$ as,

$$p(v) = 1 - \prod_{u \in Par(v)} (1 - p(u)p(u, v)).$$

The algorithm is summarized in Algorithm 1. It starts with the seed nodes and proceeds with the topological sorting order. Clearly, the algorithm has a complexity of $O(d \cdot n)$, where d is the maximum in-degree.

Algorithm 1. Single-Pass Belief Propagation (SPBP)

```
input  : $\mathcal{D}(S)$
$\sigma(S) = 0$;
foreach $v \in \mathcal{D}(S)$ do
    if $v \in S$ then
        $p(v) = 1$
    else
        $p(v) = 1 - \prod_{u \in Par(v)} (1 - p(u)p(u, v))$
    $\sigma(S) = \sigma(S) + p(v)$
output: $\sigma(S)$
```

4 DAG Construction

In general, real social networks are not DAGs (with the exception of advisor-advisee and parent-child relationship, for instance, which exhibit a natural hierarchy). To apply the BP algorithm in computing influence spread, one needs to selectively prune edges and reduce the graph to a DAG. Clearly, there are many ways to do so. The challenge is to find a DAG that approximates well the original graph in influence spread. In this section, we introduce two DAG construction algorithms, both retaining important edges where influences are likely to travel.

4.1 Localizing Influence Spread Region

One important observation in [8] is that the influence of a seed node diminishes quickly along a path away from the seed node. In other words, the "perimeter" of influence or the *influence region* of a seed node is in fact very small. One way to characterize the *influence region* of a node v is through the union of maximum influence paths defined next.

Definition 1. *(Path Propagation Probability)*

For a given path $P(u,v) = \{u_1, u_2, \ldots, u_l\}$ of length l from a vertex u to v, with $u_1 = u, u_l = v$ and $u_2, \ldots, u_{l-1}$ are intermediate vertices, define the propagation probability of the path, $p(P)$, as:

$$p(P(u,v)) = \prod_{i=1}^{l-1} p(u_1, u_{i+1}). \tag{3}$$

Definition 2. *(Maximum Influence Path)*

Denote by $\mathcal{P}(\mathcal{G}, u, v)$ the set of all paths from u to v in $\mathcal{G}$. The maximum influence path $MIP(\mathcal{G}, u, v)$ from u to v is defined as:

$$MIP(\mathcal{G}, u, v) = \arg\max_{P}\{p(P) | P \in \mathcal{P}(\mathcal{G}, u, v)\}. \tag{4}$$

Ties are broken in a predetermined and consistent way such that $MIP(\mathcal{G}, u, v)$ is always unique, and any sub-path in $MIP(\mathcal{G}, u, v)$ from x to y is also the $MIP(\mathcal{G}, x, y)$.

Definition 3. *(Maximum Influence Out-Arborescence)*

For a graph $\mathcal{G}$, an influence threshold θ, the maximum influence out-arborescence of a node $u \in V, MIOA(\mathcal{G}, u, \theta)$, is defined as:

$$MIOA(\mathcal{G}, u, \theta) = \bigcup_{v \in V, p(MIP(\mathcal{G}, u, v)) \geq \theta} MIP(\mathcal{G}, u, v). \tag{5}$$

One can think of $MIOA(\mathcal{G}, u, \theta)$ as a *local region* where u can spread its influence to. $MIOA(\mathcal{G}, u, \theta)$ can be computed by first finding the Dijkstra tree rooted at

u with edge weight $-\log(p(u, v))$ for edge (u, v), and then removing the paths whose cumulative weights are too high. By tuning the parameter θ, influence regions of different sizes can be obtained. For a single node, its MIOA is clearly a tree. For multiple seed nodes, we build upon the idea of local influence region and propose two algorithms.

4.2 Building DAGs

DAG 1. We observe that any DAG has at least one topological ordering. Conversely, given a topological ordering, if only edges going from a node of low rank to one with high rank are allowed, the resulting graph is a DAG.

To obtain the topological ordering given seed set S, we first introduce a (virtual) super root node R that is connected to all seed nodes with edge probability 1. Let $\mathcal{G}_R = (V_{\mathcal{G}_R}, E_{\mathcal{G}_R})$ where $V_{\mathcal{G}_R} = V \cup \{R\}$ and $E_{\mathcal{G}_R} = E \cup \{(R, S_k) | \forall S_k \in S\}$. We build $MIOA(\mathcal{G}_R, R, \theta)$ by calculating a Dijkstra tree from R. After removing R and its edges from $MIOA(\mathcal{G}_R, R, \theta)$, we obtain a singly connected DAG $\mathcal{D}_1 = (V_{\mathcal{D}_1}, E_{\mathcal{D}_1})$ on which BP algorithms can be directly applied and used to estimate the influence spread from S. However, $\mathcal{D}_1(\cdot)$ is very sparse (with $n - k$ edges) since many edges are removed.

We then augment $\mathcal{D}_1(\cdot)$ with additional edges. Note that $MIOA(\mathcal{G}_R, R, \theta)$ provides a topology ordering. More specifically, let the rank of node v be the sum weight of the shortest path from R, namely,

$$r(v) = \min(-\log(p(P(s, v)))), \forall s \in S. \tag{6}$$

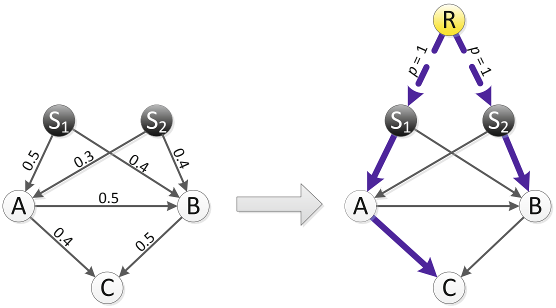
Rank grows as the node is further away from R. We include in $\mathcal{D}_1(\cdot)$ all edges in $\mathcal{G}$ whose end points are in $\mathcal{D}_1(\cdot)$ and go from a node with lower rank to one with higher rank. Clearly, the resulting graph is a DAG. The DAG constructing procedure is illustrated in Figure 3 and summarized in Algorithm 2.

Algorithm 2. Calculate $\mathcal{D}_1(S)$ from a seed set S

```
input : $\mathcal{G}, S, \theta$
Build $\mathcal{G}_R = (V_{\mathcal{G}_R}, E_{\mathcal{G}_R})$
$\mathcal{D}_1(S) = MIOA(\mathcal{G}_R, R, \theta) \backslash R$
Calculate $r(v), \forall v \in V_{\mathcal{D}_1}$ (Eq. (6))
foreach $(u, v) \in V_{\mathcal{G}_R}$ do
    if $r(u) < r(v)$ and $(u, v) \in E$ then
        $\mathcal{D}_1(S) = \mathcal{D}_1(S) \cup (u, v)$
output: $\mathcal{D}_1(S)$
```

DAG 2. In the second algorithm, we first compute the $MIOA$ from each seed node and take the union of $MIOA(\mathcal{G}, s, \theta), \forall s \in S$. Denote the resulting graph $\mathcal{D}_2(S) = (V_{\mathcal{D}_2}, E_{\mathcal{D}_2})$. Note that $\mathcal{D}_2(S)$ is not necessary a DAG as there could be circles. To break the cycles, certain edges need to be removed. We adopt a similar approach as in Algorithm 2. A node v is associated with a rank $r(v)$ as in (6). Only edges that connect a lower ranked node to higher ranked node are retained. Clearly, the resulting graph is a DAG. The approach is summarized in Algorithm 3.

The next proposition provides the relationship between DAGs constructed by Algorithm 2 and 3.



Node	S_1	S_2	A	B	C
r(Node)	0	0	0.301	0.398	0.699

Fig. 3. DAG due to Algorithm 2. S_1 and S_2 are seed nodes. Edges in $MIOA(\mathcal{G}_R, R, \theta)$ are in bold. (S_1, B), (S_2, A), (A, B), and (B, C) are added into $\mathcal{D}_1(S)$ to improve inference accuracy. $\theta = 0.0001$.

Algorithm 3. Calculate $\mathcal{D}_2(S)$ from a seed set S

```
input  : $\mathcal{G}, S, MIOA(\mathcal{G}, v, \theta), \forall v \in V$
$\mathcal{D}_2(S) = \bigcup_{\forall s \in S} MIOA(\mathcal{G}, s, \theta)$
Calculate $r(v), \forall v \in V_{\mathcal{D}_2}$ (Eq. (6))
foreach $(u, v) \in \mathcal{D}_2(S)$ do
    if $r(u) \geq r(v)$ then
        $\mathcal{D}_2(S) = \mathcal{D}_2(S) \backslash (u, v)$
output: $\mathcal{D}_2(S)$
```

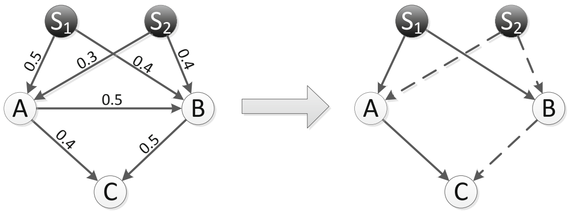
Proposition 1. *Given a fixed influence threshold θ, let $\mathcal{D}_1(\cdot) = (V_{\mathcal{D}_1}, E_{\mathcal{D}_1})$ and $\mathcal{D}_2(\cdot) = (V_{\mathcal{D}_2}, E_{\mathcal{D}_2})$ be the DAGs constructed by Algorithm 2 and Algorithm 3. Then, $V_{\mathcal{D}_1} = V_{\mathcal{D}_2}$ and $E_{\mathcal{D}_2} \subseteq E_{\mathcal{D}_1}$.*

Computation Complexity. The computation complexity of a Dijkstra tree is $O(n^2)$. When a new seed node is added, the worst cast computation time is $O(n^2)$ (if the corresponding $MIOA$ needs to be computed anew). The union operation in DAG 2 takes $O(n-1)$ time, and the edge pruning in DAG 1 and DAG 2 take $O(m)$ and $O(\min(m, k(n-1))$, respectively.

5 Accelerated Greedy Algorithm

In the original Greedy algorithm [4], in each round, a seed node with the maximum increment on influence spread is selected, namely, $v = \max_{v \in V \backslash S}(\sigma(S \cup \{v\}) - \sigma(S))$. We call $\delta_S(v) = \sigma(S \cup \{v\}) - \sigma(S)$ the spread increment of v under S. Initially, when $S = \emptyset$, $\delta_S(v) = \sigma(v)$.

To accelerate the execution of Greedy algorithm, one can try to improve on two aspects, namely, 1) limiting the set of nodes to pick from for the next



Node	S_1	S_2	A	B	C
r(Node)	0	0	0.301	0.398	0.699

Fig. 4. DAG due to Algorithm 3. S_1 and S_2 are seed nodes. $\mathcal{D}_2(S)$ is the union of $MIOA(\mathcal{G}, S_1, \theta)$ (solid edges) and $MIOA(\mathcal{G}, S_2, \theta)$ (dashed edges). $\theta = 0.0001$.

seed, and 2) reducing the complexity of computing the spread increments. CELF algorithm [5] eliminates many nodes from being evaluated. We focus on the second aspect. The proposed mechanism can be used in conjunction with CELF.

Recall in Section 4.1, we use $MIOA$ to localize the influence region of a node. Consider for now that influence from a node can only reach nodes in its $MIOA$. Then, we make the following claim.

Proposition 2. *Given the current seed set S, adding u to S will not change the spread increment of v, namely, $\delta_S(v) = \delta_{S\cup\{u\}}(v)$ if $MIOA(\mathcal{G}, u, \theta)$ and $MIOA(\mathcal{G}, v, \theta)$ have no common vertex.*

As a result of Proposition 2, each time we select a new seed, only the influence increments of nodes that have overlapping influence regions with the newly selected seed need to be re-evaluated. Formally, we define the set of Peer Seeds (PS) of a vertex $v \in V$ as follow:

$$PS(\mathcal{G}, v, \theta) = \{s \in V | MIOA(\mathcal{G}, s, \theta) \cap MIOA(\mathcal{G}, v, \theta) \neq \emptyset\} . \tag{7}$$

$PS(\mathcal{G}, v, \theta)$ can be computed efficiently just once at the beginning when all $MIOA(\mathcal{G}, v, \theta)$'s are available. To this end, we summarize the complete procedure to determine the optimal seed set in Algorithm 4.

6 Evaluation

In this section, we evaluate the performance of the proposed algorithms. Large scale social networks are used to evaluate the maximum influence spread of different algorithms. In addition to the two DAG models and two methods to compute influence spread (a total of 4 combinations DAG1–LBP, DAG1–SPBP, DAG2–LBP, and DAG2–SPBP), we make comparison with the following algorithms:

- **PMIA(θ) [8]:** a very fast heuristic that builds a tree-like structure model on which influence is spread. We set the influence threshold $\theta = 1/160$.

Algorithm 4. Accelerated Greedy Algorithm

```
input : network graph $\mathcal{G}(V, E)$ and seed set size $k$
// initialization
$S = \emptyset, \sigma_0 = 0, \theta =$ influence threshold
foreach $v \in V$ do
    build $MIOA(\mathcal{G}, v, \theta)$
    $\mathcal{D}(v) = MIOA(\mathcal{G}, v, \theta)$
    calculate $\sigma(v)$ (LBP or Algorithm 1)
    $\delta(v) = \sigma(v)$
    $\delta_{old}(v) = 0$
build $PS(\mathcal{G}, v, \theta), \forall v \in V$
// main loop
for $i = 1, \ldots, k$ do
    // select the i'th seed
    $u = \arg\max_{v \in V \setminus S}(\delta(v))$
    $S = S \cup \{u\}$
    $\sigma_0 = \sigma(S)$
    $\delta_{old}(v) = \delta(v), \forall v \in V \setminus S$
    // update incremental influence spread
    $\delta_{max} = 0$
    foreach $v \in PS(\mathcal{G}, u, \theta) \setminus S$ do
        if $\delta_{old}(v) > \delta_{max}$ then
            build $\mathcal{D}(S \cup \{v\})$ (Algorithm 2 or 3)
            calculate $\sigma(S \cup \{v\})$ (LBP or Algorithm 1)
            $\delta(v) = \sigma(S \cup \{v\}) - \sigma_0$
            if $\delta(v) > \delta_{max}$ then
                $\delta_{max} = \delta(v)$
output: selected seed set $S$
```

- **Greedy:** The Greedy approach from [4] with CELF optimization in [5]. The number of simulation rounds for each $\sigma(\cdot)$ estimation is 10,000.
- **Weighted Degree:** The simple heuristic that selects k seeds that have maximum total out-connection weight.

We do not compare with other heuristics such as SP1M, SPM [19], PageRank [20], Random, DegreeDiscountIC [12] or Betweenness centrality [21] since they have been reported in previous studies [8,4,6] to be either unscalable or have poorer performance.

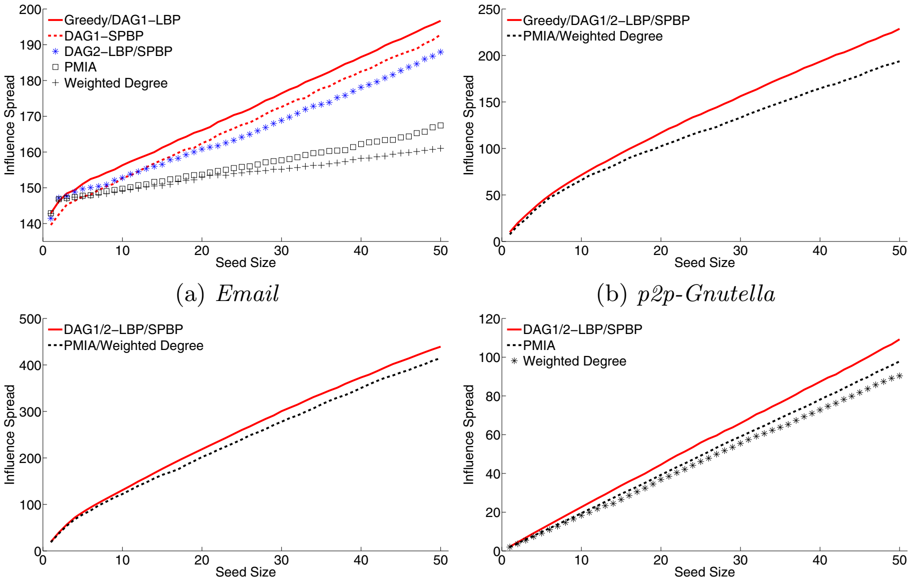
We have implemented the proposed algorithms in C++. All experiments are conducted on a workstation running Ubuntu 11.04 with an Intel Core i5 CPU and 2GB memory. In order to implement LBP algorithm, we use libDAI [22] and Boost [23] libraries. We find out through the implementation that constructing the CPT can be very costly when the in-degree of a node is high, and thus only include the parents with highest 10 influence probabilities in the factor graph. The implementation of PMIA is obtained from its authors. Note that with code optimization, the running time of our algorithms can be further reduced.

Datasets. We use four real-world network datasets from [24] and [25] to compare the experimented algorithms. Details are summarized in Table 1.

Probability Generation Model. Two models that have been used in previous work [4,8,12,6] are: 1) the WC model where $p(u, v) = 1/d(v)$ where $d(v)$ is the in-degree of v and 2) the TRIVALENCY model where $p(u, v)$ is assigned

Table 1. Network datasets

Name	*Email*	*p2p-Gnutella*	*soc-Slashdot*	*Amazon*
Nodes	447	6,301	82,168	262,111
Edges	5,731	20,777	948,464	1,234,877
Density	0.04	1e–03	1.6e–03	2.6e–05
Max Degree	195	97	5064	425
Mean Degree	25.64	6.59	23.09	9.42
Description	Email exchanged in a research lab during a year	Gnutella peer to peer network from August 2002	Slashdot social network from February 2009	Amazon product purchasing network from March 2003



(a) *Email*

(b) *p2p-Gnutella*

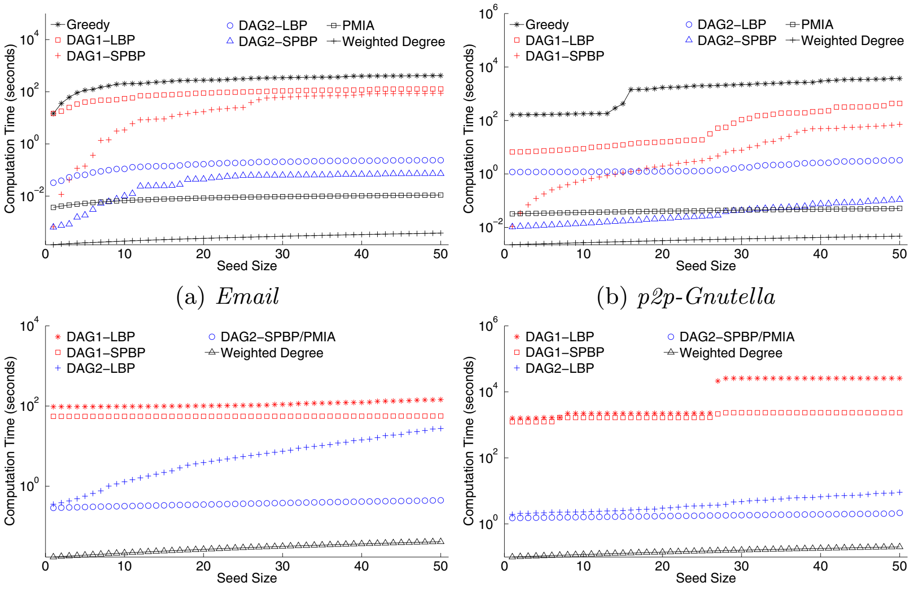
(b) *soc-Slashdot*

(c) *Amazon*

Fig. 5. Influence spread of the best seed sets on 4 datasets

a small value for any $(u, v) \in E$. We argue that both models are not truthful reflections of the probability model in practice. The WC model assign a very high probability for a connections to nodes with small number of incoming connections while the TRIVALENCY model assigns a similar probability to all edges. In our evaluation, we consider the RANDOM model where $p(u, v)$ is randomly selected in the range [0.001, 0.1].

Influence Spread and Running Time. Figure 5 shows the influence spread generated by the best seed sets in different algorithms as the seed size changes. Since Greedy does not scale with large datasets, we only run Greedy on *Email* and *p2p-Gnutella*. In this set of experiments, the influence spread from the seed



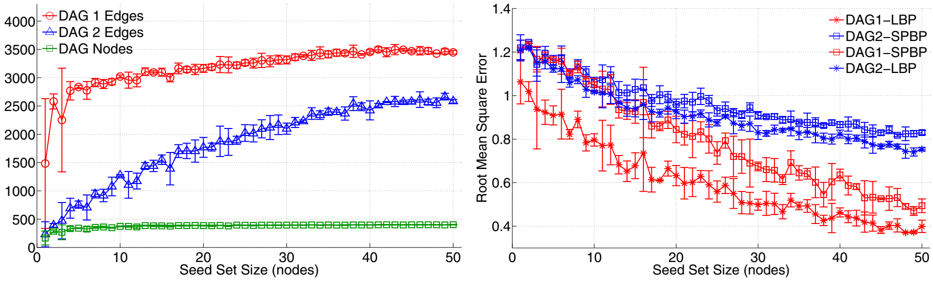
(a) *Email*

(b) *p2p-Gnutella*

(b) *soc-Slashdot*

(c) *Amazon*

Fig. 6. Computation time on 4 datasets



(a) Number of nodes and edges in DAG

(b) RMSE

Fig. 7. Size of DAGs and RMSE of activation probabilities. Results are averages of 50 runs with different seed selections and symmetric error bars indicate standard deviations.

set selected by each algorithm is determined by 10,000 rounds of Monte Carlo simulations on the original graphs.

In Figure 5(a), the performance of DAG1–LBP and Greedy (known to be within a constant ratio of the optimal) are not distinguishable (and thus are represented in one curve). The influence spread of DAG1–SPBP and DAG2–LBP/SPBP are shortly behind, all outperforming PMIA and Weighted Degree. We observe on *Email* dataset (a small but dense network) that both the structure of the DAG (DAG 1 vs. DAG 2) as well as the BP algorithm used (LBP vs. SPBP) will affect performance of the proposed methods. In contrast, as shown in Figure 5(b) – (d), the influence spreads of the four approaches DAG1/2–LBP/SPBP are identical for sparser networks, and is the same as Greedy in *p2p-Gnutella* dataset.

In terms of running time, Weighted Degree is the fastest. Among the four proposed approaches, DAG2–SPBP is the fastest, next are DAG2–LBP, DAG1–SPBP, and finally DAG1–LBP. DAG2–SPBP and PMIA have comparable order in running time with DAG2–SPBP being 30-40% slower than PMIA in most cases. Again, this may be primarily attributed to the lack of code optimization in our proposed methods.

Interestingly, influence spread on *Amazon* grows linearly with the seed size. Our result matches with that in [8]. This can be explained by the sheer scale of the network, and thus the small number of selected seeds are likely to have non-overlapping influence regions.

Comparison of the Two DAG Models. To understand the behavior of the proposed algorithms, we conduct further experiments on *Email* dataset as it gives the most performance difference between the experimented algorithms.

Figure 7 (a) gives the number of vertices and edges as the result of the two DAG models with varying size of seed sets. Since both have the same number of vertices, only one curve is shown. It it clear that DAG 1 is much bigger from DAG 2 due to the inclusion of more edges. As the seed set grows, the gap becomes smaller.

We use Root Mean Square Error (RMSE) to compare the activation probabilities on nodes. RMSE is defined as,

$$RMSE(p,p') = \sqrt{\frac{\sum_{\forall v \in V}(p'(v) - p(v))^2}{n}} / \frac{\sum_{\forall v \in V}^{n} p(v)}{n},$$

where $p'(\cdot)$ is the inferred result from the propose algorithms. The ground truth $p(\cdot)$ is determined by Monte Carlo simulations. When $p'(v) = p(v), \forall v \in V$ then $RMSE(p,p') = 0$.

Figure 7(b) shows that DAG 1 methods have smaller RMSE since they are based on a denser graph. More edges clearly help increase quality of the seed selection process. In the context of LBP vs. SPBP, LBP is slightly better since SPBP get rid of the state correlation between nodes. DAG 1 and LBP can help produce better inference result, but entails more computation complexity. The results are consistent with those in Figure 5(a).

Summary. From the conducted experiments, Weighted Degree gives the best efficiency in terms of spread/complexity. However, there are cases (*Email* dataset) in which Weight Degree performs poorly. Our proposed schemes works well in all the experimented datasets. They also offer more application flexibility: one would apply the best performed algorithm (DAG1–LBP) on static networks (e.g.: network of connections between co-workers) to identify the most influential nodes, or apply the fastest algorithm (DAG2–SPBP) on rapidly changing communities (e.g.: network of connections between people in a social group) to obtain immediate result.

7 Conclusion

In this paper, we considered the IM problem on social networks where the objective is to find a set k of nodes that can maximize the influence spread. We established the linkage between influence spread computation and BP on a Bayesian network. With 2 DAG models and 2 BP algorithms, 4 methods are proposed offering the flexibility between computation time and accuracy. Simulations using real-world social network graphs show that the proposed schemes achieve higher influence spread compared to the best known solutions. Interestingly, DAG 2 model, although being much smaller than DAG 1, gives a good approximation result that is comparable to DAG 1 with only a marginal computation cost. Result also exhibits the dependency of algorithm performance over the experimented network. Thus suggesting an interesting research direction to study the impact of graph structure in IM problem.

Acknowledgment. This work is funded in part by the National Science Foundation under grants CNS-1117560 and CNS-0832084.

References

1. Dignan, L.: HP's touchpad fire sale: The fallout (2011), http://www.zdnet.com/blog/btl/hps-touchpad-fire-sale-the-fallout/55594
2. Domingos, P., Richardson, M.: Mining the network value of customers. In: Procs. of KDD 2001, pp. 57–66. ACM, New York (2001)
3. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: Proceedings of KDD 2002, pp. 61–70. ACM, New York (2002)
4. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD, KDD 2003, pp. 137–146. ACM, New York (2003)
5. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD, KDD 2007, pp. 420–429. ACM, New York (2007)
6. Goyal, A., Lu, W., Lakshmanan, L.V.: Celf++: optimizing the greedy algorithm for influence maximization in social networks. In: Proc. of WWW 2011, pp. 47–48. ACM, New York (2011)
7. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 199–208. ACM, New York (2009)
8. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of KDD 2010, pp. 1029–1038. ACM, New York (2010)

9. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: Proceedings of ICDM 2010, pp. 88–97. IEEE Computer Society, Washington, DC (2010)
10. Even-Dar, E., Shapira, A.: A note on maximizing the spread of influence in social networks. Inf. Process. Lett. 111, 184–187 (2011)
11. Sylvester, J.: Maximizing diffusion on dynamic social networks. Science (2009)
12. Chen, W., Collins, A., Cummings, R., Ke, T., Liu, Z., Rincon, D., Sun, X., Wang, Y., Wei, W., Yuan, Y.: Influence maximization in social networks when negative opinions emerge and propagate. In: SDM, pp. 379–390. SIAM/Omnipress (2011)
13. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations, San Francisco, CA, USA, pp. 239–269 (2003)
14. Pearl, J.: Reverend Bayes on inference engines: A distributed hierarchical approach. In: Proceedings of the AAAI 1982, Pittsburgh, PA, pp. 133–136 (1982)
15. Lauritzen, S.L., Spiegelhalter, D.J.: Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. Journal of the Royal Statistical Society. Series B (Methodological) 50(2) (1988)
16. Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy Belief Propagation for Approximate Inference: An Empirical Study. In: Procs. of Uncertainty in AI, pp. 467–475 (1999)
17. Frey, B.J., Koetter, R., Petrovic, N.: Very loopy belief propagation for unwrapping phase images, vol. 14, pp. 737–743. MIT Press (2001)
18. McEliece, R., MacKay, D., Cheng, J.F.: Turbo decoding as an instance of pearl's ldquo;belief propagation rdquo; algorithm. IEEE Journal on Selected Areas in Communications 16(2), 140–152 (1998)
19. Kimura, M., Saito, K.: Tractable Models for Information Diffusion in Social Networks. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 259–271. Springer, Heidelberg (2006)
20. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. 30, 107–117 (1998)
21. Freeman, L.: Centrality in social networks conceptual clarification. Social Networks 1(3), 215–239 (1979)
22. Mooij, J.M.: libDAI: A free and open source C++ library for discrete approximate inference in graphical models. Journal of Machine Learning Research 11, 2169–2173 (2010)
23. Boost.org: Boost c++ libraries, http://www.boost.org/
24. Leskovec, J.: Stanford large network dataset collection, http://snap.stanford.edu/data/
25. Wiki, I.: Social network generation, http://www.infovis-wiki.net/index.php/Social_Network_Generation

Location Affiliation Networks: Bonding Social and Spatial Information

Konstantinos Pelechrinis and Prashant Krishnamurthy

School of Information Sciences
University of Pittsburgh
{kpele,prashant}@sis.pitt.edu

Abstract. Location-based social networks (LBSNs) have recently attracted a lot of attention due to the number of novel services they can offer. Prior work on analysis of LBSNs has mainly focused on the social part of these systems. Even though it is important to know how different the structure of the social graph of an LBSN is as compared to the friendship-based social networks (SNs), it raises the interesting question of what kinds of linkages exist between locations and friendships. The main problem we are investigating is to identify such connections between the social and the spatial planes of an LBSN. In particular, in this paper we focus on answering the following general question "What are the bonds between the social and spatial information in an LBSN and what are the metrics that can reveal them?" In order to tackle this problem, we employ the idea of *affiliation networks*. Analyzing a dataset from a specific LBSN (Gowalla), we make two main interesting observations; (i) the social network exhibits *signs of homophily* with regards to the "places/venues" visited by the users, and (ii) the "nature" of the visited venues that are common to users is powerful and informative in revealing the social/spatial linkages. We further show that the "entropy" (or diversity) of a venue can be used to better connect spatial information with the existing social relations. The entropy records the diversity of a venue and requires only location history of users (it does not need temporal history). Finally, we provide a simple application of our findings for predicting existing friendship relations based on users' historic spatial information. We show that even with simple unsupervised learning models we can achieve significant improvement in prediction when we consider features that capture the "nature" of the venue as compared to the case where only apparent properties of the location history are used (e.g., number of common visits).

1 Introduction

During the last few years, boosted by advancements in mobile handheld devices (e.g., smartphones), a new class of digital social networks, namely location-based social networks (LBSNs), has emerged. It is now possible to bring into the equation of online social networks (OSNs) another dimension, that of ***location***, due to the significantly improved ability of mobile devices to accurately estimate their position or location. The underlying communities not only have social ties (e.g., friendship) and/or interests in common (e.g., sports), but they are also "connected" with regards to their geographic

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 531–547, 2012.

locations (often mapped into "venues" as described later). In other words, LBSNs bond the online and physical social ties through location information.

This bond can enable a number of novel, convenient, and appealing services making LBSNs popular. People can now track their children's locations. By tracking friends, applications such as better coordination for scheduled meetings can be enabled. Applications can also include exploring new places through a list of venues that are within the proximity of the current location. This list can now be accompanied by tips and recommendations from people/friends that have visited these places. Even simply the number of people that have visited a locale in the past or are present at the moment might be helpful and informative. Other systems can also offer Groupon-like deals, providing additional monetary incentives for someone to adopt their usage. A recent study has also shown that "gaming" aspects of LBSNs form an important motivation for people to start using them [12].

With LBSNs becoming prevalent, it becomes critical to comprehend and discriminate the types of knowledge we can obtain from the bond between locations and social ties. For example, what correlations exist between users' spatial trails and their social behaviors as expressed through their friendships and do the spatial trails provide any information about social ties? Our primary objective in this work is to identify the existing correlations and the metrics that can best capture them. Using the knowledge we obtain from our study we further examine *whether we can use these correlations and metrics to infer social information* **only** *from users' locations*. Going forward this can stimulate our ability to deconstruct the interplay between the social and the spatial information plane and apply it to new applications.

Interactions in an LBSN: An LBSN has two distinct components; a social network and a location log for each member. The social part of the system resembles any other existing online social network, where friendships are declared and people can interact with their friends. What differentiates LBSNs from other OSNs are the type of interactions that are feasible between the members of the network. The main feature of this interaction is location sharing. While the "visible" interactions in a traditional OSN are restricted to the virtual world, we can observe interactions within an LBSN in the physical world as well. This is especially important for our study since it can shed light on patterns that are otherwise difficult to identify.

Location sharing can be realized either through continuous tracking, in the form of a temporal latitude/longitude trajectory (e.g., Loopt - see Figure 1) or via "check-ins", where users announce their presence in a place or venue at their convenience (e.g., Gowalla, Foursquare etc. - see Figure 2). Clearly, the second approach, where location is tagged with semantic information as compared to a flat geographic trajectory, offers a richer set of information, but with coarse location granularity. All major LBSNs follow this latter approach and consequently, in this work we consider systems in which spatial information is created via check-ins. We note here that using "check-in" history can be challenging since fine grained temporal information is absent (e.g., users do not "check-out" etc.).

Hence, we now have two types of information – the social ties between members and check-ins of members of the LBSN. To analyze socio-spatial interactions within an LBSN, we model it as an "affiliation network", where the members are nodes of one

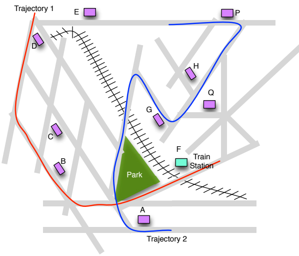


Fig. 1. Trajectory-based LBSN

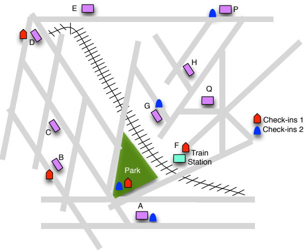


Fig. 2. Check-in based LBSN

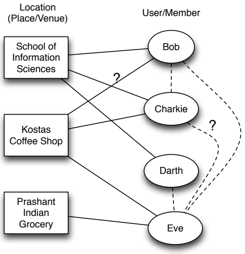


Fig. 3. LBSN as affiliation network

type and venues/places are nodes of the second type (see Figure 3). Using a dataset from Gowalla [2], we analyze how the *number* and *type* of users' common affiliations (as measured through the number of common locales visited by them) are related to the affinities in the underlying social graph. The main contributions of our study can be summarized as follows:

- We identify clear signs of location homophily, that is, members of the LBSN that are friends are more *similar* compared to those that are non-friends. "Similarity" here refers to the **percentage** of visited places that are common between two users (formally defined later).
- While simply the number of common places visited by two users does not provide rich social knowledge, the user similarity as well as the "type" of their common venues is a very descriptive feature.

Using the affiliation network model we are able to define the clustering coefficient (cc) of a venue, which captures the type and diversity of the latter. As we will see later, this cc has a strong correlation with the social relations in the graph; exactly what we are looking for! However, its computation utilizes knowledge from the friendship graph, resulting in the problem of circular reasoning. Hence, we examine other metrics, and in particular we show that the entropy of a venue is very informative and helpful for dealing with our problem.

Finally, we investigate the importance of the different features we consider through simple unsupervised friendship prediction models. In particular, we seek to infer the existing affinity relations using *only* the users' location history. Our evaluations reveal that features that account for the type of a venue, can significantly improve the estimations as compared to features that consider all venues equal.

Scope of Our Study: We would like to emphasize that our work is a study of the interplay between the social and spatial information present in an LBSN. Even though this connection can enable many new applications, such as location prediction, this study is not focused on any specific one of them. Despite the fact that we examine some simple friendship inference models utilizing our findings, our objective in this study is **not** to provide a social affinity classifier but to provide insights into the value of the location information present in an LBSN and its strength for predicting social ties. For instance, the relation between spatial and social data can have significant implications on users' privacy. Privacy policies that avoid information leakage from one component of the network to the other should be designed and be in place. We believe that this work can stimulate further research and enhance existing – or even enable new – functionalities within an LBSN.

The rest of the paper is organized as follows. Section 2 discusses work related to our study, while Section 3 describes our affiliation network model for an LBSN and the dataset. Section 4 briefly presents the analysis of the social graph of Gowalla. Our study on the relation between users' location information and their social ties is presented in Section 5. Finally, Section 6 presents our friendship inference model, while Section 7 concludes our work.

2 Related Work

There is a set of studies that examine the structural properties of existing LBSNs. In this context, structure refers not only to the properties of the social network graph (as in OSNs) but also to the location component (e.g., physical distance to friends, time and type of check-ins etc.). For instance, Cheng *et al.* [1] use data from Foursquare to examine (i) the spatio-temporal properties of users' check-ins, as well as (ii) their mobility patterns. Similarly, Noulas *et. al.* [16] study the spatio-temporal properties of users activities as captured through the inter-checkin times and the inter-checkin distances. They further identify universal features for human urban mobility [15]. In alignment, Cho *et al.* [2] use cell phone location and LBSN data to understand the laws dictating human mobility. Li and Chen [10] analyze data from Brightkite and after providing the structural properties of the underlying social graph they try to identify correlations between different user's profile features, activity updates, and mobility patterns.

The majority of these studies deal explicitly either with the social part of the system or the location component. Scellato *et al.* [18] try to use information from both components to identify the relation between friendship and geographic distance using data from 3 different LBSNs (Gowalla, Foursquare and Brightkite). They find that the socio-spatial structure of these systems cannot be explained by only geographic factors or only social mechanisms. In addition, there exist a few studies in the literature that examine and analyze the location data present in the system with the goal of revealing undirect, hidden information. Noulas *et al.* [17] obtain a static snapshop of Foursquare in order to analyze the activity in different neighborhoods of London and New York, while Ye *et al.* [22] exploit social and spatial characteristics of LBSNs for location recommendation.

None of the aforementioned works however, study the relation between the location trace of a user and his social relationships. They are all mainly focused on identifying patterns either in the social or in the spatial component of an LBSN. Eagle *et al.* [5] [6], as well as Li *et al.* [11], have developed measures to quantify similarity of users based on their mobility. This similarity can be later used to infer the social structure of the users. They are focused on "co-location instances," that is, situations where two users are at the same place at the same time. However, given the fact that co-locations between people can happen accidentally, especially in urban areas [13], simply accounting for the number of co-existences can be expected to not be very accurate. Recently, Wang *et al.* [20] using mobile phone data have identified a positive corelation between mobile homophily, network proximity and social tie strength.

The work that is closer to ours is the study by Cranshaw *et al.* [4]. In particular, they introduce the notion of a location's "entropy," which captures its *diversity* with regards to the people visiting it. Using a small scale dataset of location trajectories obtained from 397 users of Locaccino the authors infer co-locations between users. They examine the relation between features such as the intensity and duration of co-locations between people, the diversity of these co-locations, and the users' mobility regularities, with the social structure of these users. The latter is obtained through their Facebook accounts. They apply supervised learning classifiers on a set of 16 features to obtain the structure. Scellato *et al.* [19] took one step further and use location information in order to improve friend recommendations. They are focused on the temporal evolution of the social graph and they utilize a combination of information drawn from both the social and location component to improve friend recommendations. On the contrary, we are focused on a static snapshot of the network and we are looking into relations between the two information planes of the system.

To further differentiate our work, the location information that Cranshaw *et al.* [4] consider includes the *complete trajectory* of users, from which features such as the duration of a co-location can be inferred. Nevertheless, despite the fact that such data include fine-grained spatio-temporal information, note here that such data capture the location of the device and not necessarily that of the user to begin with (a problem identified in [4] as well). On the contrary, in an LBSN such as Gowalla or Foursquare, people check-in to spots declaring their actual presence in a physical location. However, users do not "check-out", making it challenging, if not impossible, to obtain detailed spatio-temporal information (e.g., co-location duration, even actual co-location at all). Yet, as we show later, this information is promising in its ability to tie the spatial and social plane of LBSNs. To summarize, in contrast to the work presented in [4] we are only using the check-in history of users without considering information related to the actual co-locations of users (i.e., temporal information).

3 Location Affiliation Network

In this section we will briefly describe the data set and affiliation network model for the LBSNs used in this paper.

Gowalla Dataset: The dataset consists of 6,442,892 public check-in data performed by 196,591 Gowalla users in 647,923 distinct places, during the period between February 2009 and October 2010. Gowalla users also participate in a friendship network with reciprocal relations, which consists of 950,327 links. The public dataset [2] includes only an ID for the spot of the check-in. We have further crawled the web in order to obtain a mapping between this id and the actual locale (or "spot" in the terminology of Gowalla[1]). Note here that since the acquisition of Gowalla from Facebook, its public

[1] We will use the terms locale, place, venue, spot and affiliation interchangeably.

website is offline. However, we were able to obtain a subset of the required information through the Internet Archive Wayback Machine and Google Cache.

Affiliation Network: Social relations can be formed due to a variety of reasons. For instance, it has been observed that people tend to relate to others with similar characteristics/interests (**homophily**) [9]. When we refer to immutable characteristics it is clear that the main reason behind homophily is the mechanism of **selection** [8]. For instance, people prefer in general to socialize with people of the same nationality. However, when we consider mutable characteristics (e.g., political views) it is not clear whether selection or **social influence** [7] leads to homophily. In the latter case, friendships were first created and then people influenced each other and became similar.

Based on the above, link creation is affected by contextual factors related to the *similarity* between the users. This similarity can refer to characteristics, activities, or behaviors. However, the representation of a social network as a flat affinity graph is not capable of capturing these surrounding contexts. Affiliation networks integrate "focal points" (*foci*) of social interactions with the pure social graph [14]. An affiliation network is essentially a bipartite graph with two sets of nodes, S and F. S is the set of nodes that represents the members/users of the network, while F represents the activities (affiliations or foci) into which users engage. An edge $\{(s, f) : s \in S \wedge f \in F\}$ exists, iff s is participating in focus f. Two users u and v are said to be affiliated if they participate in the same activity f. Hence, the affiliation network becomes the layer on which the actual social network is created. As Watts states, "without any affiliations, the chance that two people will be connected is negligible" [21].

If we further connect members of S based on their social relations, we obtain a *social-affiliation* network (see Figure 3). Using the latter we can analyze the co-evolution of both the social and the affiliation networks. A new friendship might be created due to a common friend (**triadic closure**), or due to a common affiliation (**focal closure**). Furthermore, a new affiliation can be created due to a friend already affiliated with it (**membership closure**). Focal closure is an artifact of the selection process, while membership closure is a type of social influence. In the LBSNs that we consider, the set F consists of the locations/places that people in S can check-in. An affiliation edge is created as long as a user has checked-in a specific spot. For instance, in Figure 3, Bob has checked-into the "School of Information Sciences" and hence there is an affiliation edge that connects him with the corresponding focus.

Before presenting our analysis, we would like to reiterate that data obtained from a system like Gowalla cannot provide fine-grained spatio-temporal information as in [4]. Hence, many of the detailed features used in that study are not available through our dataset. However, even if we do not know whether two friends' affiliations were created simultaneously (i.e., co-location) or with a time lag, their common affiliation is an indicator of a possible relation, and hence a socio-spatial tie. This can be attributed to the fact that both selection and social influence dictate that "friends" will tend to have a number of common affiliations. The difference is that in the former case these affiliations were present when the friendship was formed (and they might have actually caused this social relation to be formed), while in the latter the opposite occurred.

Using the terminology introduced to restate our main objective, we seek to identify patterns/correlations in the social-affiliation network that can reveal ties between the

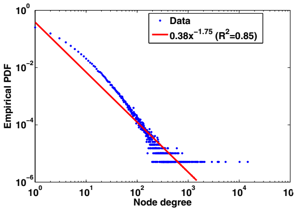


Fig. 4. Node degree distribution

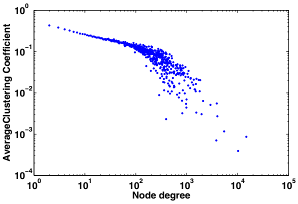


Fig. 5. Clustering coefficient vs node degree

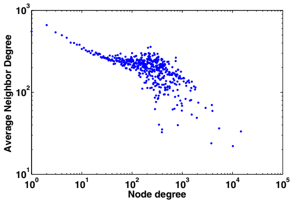


Fig. 6. Average neighbor degree

pure social and pure affiliation network. Note again that when we have a static snapshot of a network, we do not know whether an affiliation or a friendship was created first. However, once again, the actual underlying mechanism that caused the closure between two users and a group is irrelevant and what matters is the existence of a *triangle* that connects users and locales.

4 Social Network Analysis

In this section, we will briefly present our analysis of the social (friendship) graph of Gowalla. There exist similar efforts in the literature for other online social networks and hence this is not the main focus of our study. However, we are presenting these results for completeness.

Degree Distribution: First, we examine the degree distribution of the network. We compute the empirical probability density function of a user's degree (Figure 4). The degree distribution of Gowalla users follows a power law. This has been found to be true for other social networks as well [21], and implies that the majority of the users have very few friends (even none), while very few users have many friends. Formally put, the probability of a node u having a degree of x obeys the following rule:

$$Pr\{deg_u = x\} \propto \frac{1}{x^\alpha} \tag{1}$$

In Figure 4 we have also fit a power law curve, with an exponent of $\alpha = 1.75$. The high R^2 value ($R^2 = 0.85$) implies a good fit, which further supports the validity of the underlying power law. The average node degree is also computed to be 9.66.

Clustering Coefficient: Clustering coefficient (cc for short) is tightly related to the notion of triadic closure. In particular, the clustering coefficient of Bob is an indicator of how many triangles he participates in. Given that the clustering coefficient of Bob is the ratio between the pair of his friends that are friends with each other, over all the possible pairs between them, it is evident that it needs to be presented as a function of the node degree. Figure 5 presents the (average) clustering coefficient of a user with respect to his degree. As we can see, Gowalla users in general exhibit high coefficients, with the average clustering coefficient being equal to 0.237. This means that on average there is a 23.7% probability that two randomly selected friends of Bob will also be friends. This fairly high clustering coefficient, in conjunction with the small average path length, are strong indications that the social component of Gowalla is a *small world network*.

Average Neighbor Degree: The average neighbor degree $d(k)$ is a summary statistic of the joint degree distribution. It is simply the average neighbor degree of the (average) k-degree node. Figure 6 depicts $d(k)$. As we can see there is no *preference* of users to connect to peers with dissimilar or similar degrees. This can be also captured from the assortativity coefficient of the graph which is close to 0 (-0.029). The slight negative value indicates a very small degree of disassortativity; there are slightly more links connecting nodes of dissimilar degrees.

5 The Richness of Location Information

In this section we will analyze the structure of the spatial component of the LBSN. Our goal is to identify existing correlations, if any, between location information or spatial behavior (represented by the affiliations or checkins at various venues) and the social structure of the network. We are mainly interested in both direct and indirect information derived from location history. For instance, the number of common venues visited by users belongs to the first category. However, information related to the *nature* of the venue is not directly observable from the trails, but it can be inferred.

Location-Based User Similarity: As previously mentioned, homophily is a phenomenon that is very often observed in social networks. For instance, empirical studies have shown that teenagers tend to create friendships with other teenagers with similar scholastic performance and delinquent behavior (e.g., drug use) [8]. In another study, Christakis and Fowler [3] found that social relationships exhibit signs of homophily with regards to the obesity level, in a social network consisting of approximately 12,000 people. Regardless, of the reasons behind homophily, awareness of its existence can help towards revealing possible social links by observations of people's characteristics and/or behaviors and vice versa. To the best of our knowledge there is no study to date that examines homophily related to the locations visited by people. In what follows, we take a first approach to this problem. Our analysis indicates that there are ***signs*** of homophily with regards to the spatial behavior of the users. However, we would like to particularly emphasize that we do not claim to have completely answered this question. Identifying homophily in a social network is an extremely challenging task, which would require the study of longitudinal data, possibly from different networks, on a much larger scale. We hope though, that our work will encourage further research on this topic, which becomes increasingly important nowadays more than ever, with the prevalence of mobile devices with positioning capabilities and the availability of huge volumes of spatial data.

Let us define L_c, to be the set of venues that user c has checked-in. Then we define the similarity $s(u,v)$ between u and v (who have each visited at least one venue) as the following ratio:

$$s(u,v) = \frac{|L_u \cap L_v|}{|L_u \cup L_v|} \tag{2}$$

The numerator is the number of common places visited by the two users, while the denominator is the number of places visited by at least one of them. The above ratio is

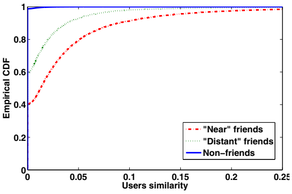


Fig. 7. Empirical CDF for user similarity

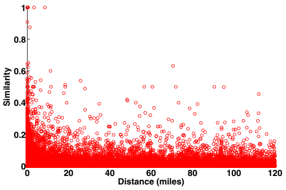


Fig. 8. Near by friends' similarity vs distance

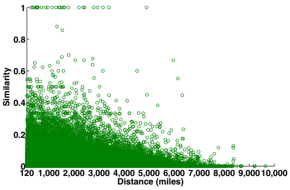


Fig. 9. Distant friends' similarity vs distance

the Jaccard similarity coefficient. We have calculated this ratio for pairs of users that are friends and pairs of users that are not friends. We have also further distinguished the pairs of users as being in geographic proximity or not, based on their "home" locations. We have set up a threshold of 120 miles for defining pairs that are "nearby" or "distant".

Figure 7 presents the similarity for three classes of pairs; nearby pairs of friends, distant pairs of friends and nearby pairs of non-friends. Clearly, friends that reside in geographic proximity to each other, have the highest similarity scores. Approximately, 35% of them have coefficients larger than 5%, which means that 5% of the places they have visited are common. This number might seem small, but it is actually fairly large if we think of the number of places we visit every day. The importance of this value becomes even more clear when we see the similarity index for nearby pairs of non-friends, which is practically 0 even though they are in geographic proximity! Note here that, even friends that are far away, exhibit similarity much higher than nearby pairs of users that are not friends. This is an important result since it implies evidence of homophily in the network with regards to the places visited. Users that are friends will visit the same spots, even if their home locations are far apart. Users that are not friends, even if they are in proximity (e.g., in the same city) are unlikely to visit the same places.

Note here that in the definition of users similarity (Equation 2), we have not considered any temporal information. We consider all common venues that have been visited by two users, regardless of whether they visited them at the same time or not. The reason for this, is that people can be *similar* in ways that do not dictate co-location. For instance, if the selection process is responsible for the high similarity values, people with the same affiliations (captured from the places they visit) will tend to create friendships. On the other hand, if social influence is responsible for the high similarity coefficient, people will tend to visit places that they have heard from their friends (however, not necessarily with them). Hence, the Jaccard similarity index can be quite helpful in bonding social and location information, even without the fine grained temporal information used in previous works. The importance of this finding is that it indicates that the characteristics of location information can be substantially different between friends and non-friends.

Next, Figure 8 shows the similarity values for nearby friends as a function of distance between home locations. As we can see, distance does not appear to have any effect on the similarity for these users. A slight decrease of the (average) coefficient can be observed, but it is not significant. However, distance appears to be critical for friends that live far apart (as one might have expected). As we see in Figure 9, after some

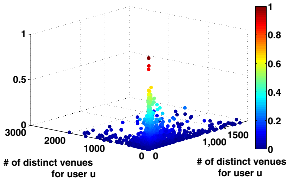


Fig. 10. Similarity as a function of the users distinct affiliations

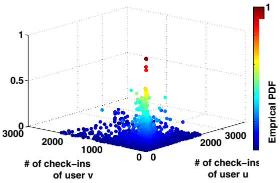


Fig. 11. Similarity as a function of the users checkins

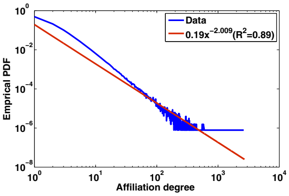


Fig. 12. Affiliation degree distribution

distance (approximately 2,500 miles) the similarity values are drastically reduced. Bob will have less opportunities to "follow" the trails of his friend Alice if she lives far away. Due to space limitations, we do not plot the similarity indexes for the pairs of non-friends, but as one can see from the cumulative distribution function, the majority of these points lay slightly above or on the $y = 0$ line irrespective of the distance.

Figures 10 and 11 present the similarity of two near friends as a function of the number of their distinct affiliations and their check-in counts respectively. Even though our data consist of a static snapshot, this figure can be seen as an "emulation" of the temporal evolution of the similarity value of two nearby friends. Higher levels of activity represent later points in time, when users have been using the system for longer periods and thus, have more affiliations and check-ins. Further, the similarity scores take their maximum values for pairs of users with low levels of activity (i.e., small number of affiliations and check-ins). Based on the above "temporal emulation" this corresponds to early stages of system adoption. This behavior can be attributed to the fact that the denominator of Equation 2 increases faster as compared to the enumerator. One possible reason that can cause this is as follows. Consider Bob and his friend Alice. Bob will hear from Alice about a few places and he will tend to visit some of them, increasing the numerator of $s(Bob, Alice)$. However, he will hear about other spots from his friends Jack and Jill (who might have no relation with Alice). Hence, he might be tempted to visit some of these spots as well, increasing the denominator faster and overall reducing the Jaccard index as his (and Alice's) level of activity increases. Therefore, even friends might exhibit low(er) similarity scores after some time, and for this reason the absolute number of common foci might be a more robust metric over longer time spans. Later in Section 6 we will use $|L_u \cap L_v|$ as the feature of our baseline social link prediction. The similarity of a pair of users, balances the above quantity, by considering the activity of both users. Such a balancing, in essence, captures the diversity of the two users; the larger the denominator, the more places they visit (more diverse user pair). As we will see in our evaluations, this balancing can provide better connection between social and spatial information. We note here that similarity values of non-friends are small regardless their level of activity (omitted due to space limitations).

Focal Closure: In an affiliation network, the foci are also nodes of the network. Hence, we can define metrics such as the degree distribution and the clustering coefficient for venues. The degree of an affiliation-node l (i.e., a venue) is the number of distinct users that have visited it. In other words, it is the number of user-affiliation links whose one endpoint is l. Figure 12 depicts the affiliation degree distribution, which as we can see

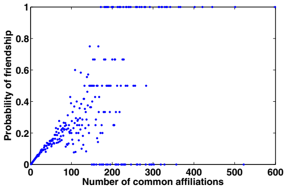


Fig. 13. No clear correlation exists between # of common foci and friendship probability

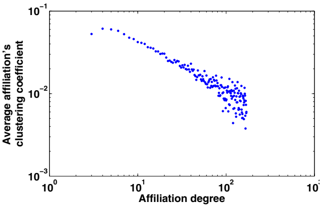


Fig. 14. Venues with lower degree tend to have lower clustering coefficient as well

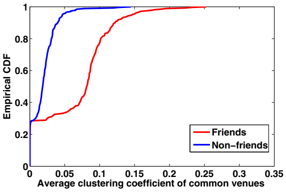


Fig. 15. Common venues of friends have larger clustering coefficient as compared to non-friends

Table 1. Top and bottom 5 venues based on their degree

Top-5 spots	SFO airport	Stockholm Central Station	AUS airport	DFW airport	LAX airport
Bottom-5 spots	"Room"	Farmer's Market	Gas station	Apparel store	Convenient store

follows a power law as well, with exponent $\alpha = 2.01$ ($R^2 = 0.89$). There are a few places with many visitors, while there are many venues with few visitors. The average focus degree is 3.11. Table 1 has the top and bottom 5 venues with regards to their degree. Note here that the bottom 5 venues were randomly selected since there are many venues (almost half) with degree of 1. The top spots are all major transportation hubs (airports or train stations), while the less popular places are more *localized*/personal venues (e.g., "room", which most probably refers to a home, office etc.). The top degree spots are expected to increase the number of common affiliations for many users; the higher the degree of a locale more user pairs will exhibit an increased number of common foci. However, as one can imagine, common affiliations such as a big airport happen rather randomly than due to actual similarity. On the contrary, if Bob and Jack have a local food joint as common affiliation it is highly possible that this is due to their similarity (e.g., they have the same gastronomical preferences).

If our above claim does not hold and all affiliations are *equal* one should expect that the more common foci two people have, the more probable it is for them to be friends. However, our data indicate that this is not the case in an LBSN. Figure 13 presents the friendship probability between a pair of users with respect to the number of common venues visited by them. As we can see there is no clear connection between the two quantities. For small values of common venues there seems to be a linear relationship, but as the number of common affiliations increases, there is little (if any) correlation. In particular, for a number of common venues smaller than 100, the correlation coefficient is fairly high (0.61). However, for larger number of common venues, this coefficients drops to just 0.2. This further supports our previous claim, that the actual affiliation, rather than just the number of the common affiliations, plays an important role in predicting the social relations.

In order to further examine the role of the type of a venue on the social relationships we examine the clustering coefficient of a focus. Let us consider venue l which has a degree of $k > 1$. All the possible social links between the users affiliated with l are

$k(k-1)/2$. If n of them exist then the clustering coefficient, $CC(l)$ is defined as:

$$CC(l) = \frac{n}{k(k-1)/2} \tag{3}$$

This clustering coefficient captures the nature of the place in many ways. It expresses how tightly connected are the people that visit this venue. The higher the cc is, the more connected are the people affiliated with it.

Figure 14 shows the clustering coefficient as a function of the venue's degree. As we can see venues with lower degree have a higher average clustering coefficient. This is a sign that venues with lower degree might venture socialization. Delving more into this issue we present in Figure 15 the CDF of the average clustering coefficient of the common venues for user pairs that are friends and those who are not. For friends, the average clustering coefficient of their common affiliations is much higher (mean value is 0.068) as compared to those of non friends (mean value is 0.019). Finally, Figure 16 plots the probability of friendship between two users as a function of the average clustering coefficient of their common spots. As we see there is a clear positive correlation between the two quantities, which is also revealed from the high correlation coefficient between the two variables (calculcated equal to 0.89). The higher the average cc of the common foci, the larger the friendship probability.

Especially, the last result clearly indicates that the actual *nature* of the venue plays an important role to whether affiliated users are related through a friendship or not. On the one hand, places with high clustering coefficient, attract sets of people that are more tightly connected in the social plane. In addition these sets are usually small, if we recall the connection between affiliation cc and affiliation degree. On the other hand, spots with low clustering coefficient attract many people that are not socially related, just because these places have special features (e.g., large hub-airports, train stations, supermarkets etc.). One could arguably compute the average cc of the common affiliation of two people and find the probability of friendship through a simple linear regression model (Figure 16).

However, there is a problem with the above approach. In order to calculate the average cc of a venue, the social relationships need to be known! Hence, the cc does not provide an **independent** socio-spatial information linkage. Therefore, we need to find a feature of the affiliations, that (i) captures the nature of the venue, (ii) does not require the knowledge social relationships in order to be computed and (iii) is correlated with the friendship probability. This feature is the affiliation's entropy as we will describe in what follows.

The entropy of a place: Cranshaw *et al.* [4] were the first to introduce the notion of entropy of a location as a measure of its diversity. If $P_l(u)$ is the fraction of check-ins in affiliation l contributed by user u, then the entropy of l is given by:

$$e(l) = -\sum_{u:u\in S\wedge P_l(u)>0} P_l(u)log(P_l(u)) \tag{4}$$

From Equation 4 we can see that when a place is visited by many people in equal (and thus, small) proportions, its entropy will be high. In other words, high entropy corresponds to places like airports that exhibit large diversity. On the other hand, when

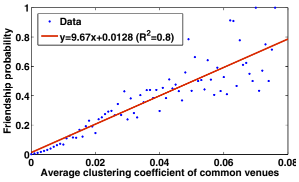


Fig. 16. Positive orrelation between avg cc of common venues and friendship probability

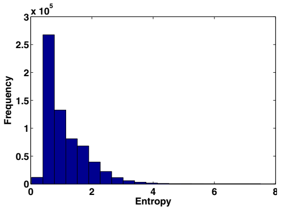


Fig. 17. The majority of the venus exhibit low entropy

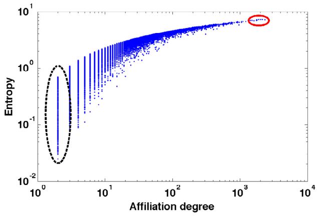


Fig. 18. The entropy of a spot increases with its degree

the mass of $P_l(u)$ is concentrated only to a few people, the diversity in this location is small and so is the entropy.

Figure 17 depicts a histogram of the entropy values for all the venues in our dataset. As we see most of the venues have small entropy values, while there are some that exhibit high entropy values. It is interesting to see that there is an increasing trend of the entropy of a place with its degree (Figure 18). Furthermore, the top-5 degree places are also the top-5 entropy places (with different ranking) as it can be seen in the red solid ellipse. A number of (the many) bottom-5 degree places are still bottom-5 entropy places. However, if we notice more carefully in the dashed, black ellipse in Figure 18, some venues with the lowest degree, do not exhibit the lowest entropy (although still smaller than 1).

Previously, we observed that there is a positive correlation between the average clustering coefficient of the common venues of two users and their friendship probability. To examine whether entropy is a good candidate for a similar correlation, we first examine its relation to cc. Figure 19 depicts the entropy of a venue as a function of its clustering coefficient. As we can see, the entropy tends to be lower as the clustering coefficient increases. High entropy translates to more random co-visits to the venue, and therefore a lower clustering coefficient. Hence, there appears to be a negative relation between these two measures (we expect a similar negative relation between the average entropy of common venues and friendship probability).

Since entropy appears to have similar characteristics with the affiliation clustering coefficient we want to further examine its ability to bond affiliation and social information. In Figure 13 we identified that the number of common affiliations is not very useful in terms of inferring the existing social relations especially when the number of common affiliations is growing (e.g., $>$ 100). We seek to further examine if we can obtain any additional knowledge by utilizing the information about the entropy. Using the same data, we consider pairs of users that have the same number of common foci. We divide them into two categories, friends and non-friends. For each one of these categories we compute the average entropy of the common spots visited and we plot the results in Figure 20. It is clear now that the average entropy of the common affiliations for the case of pairs of friends is indeed lower compared to the case of non-friends and appears to be a good candidate for bridging the social and spatial components of an LBSN.

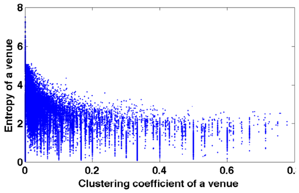


Fig. 19. Venues with higher clustering coefficient tend to have lower entropy

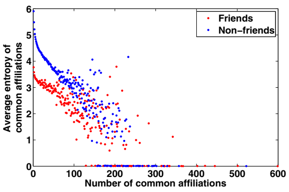


Fig. 20. The common affiliations of friends exhibit lower (average) entropy

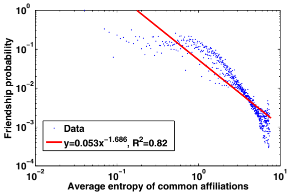


Fig. 21. Negative correlation between the avg entropy of common foci and friendship probability

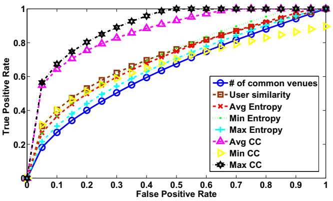
Next, we compute the average friendship probability between two users as a function of the average entropy of their common affiliations. The results presented in Figure 21 are promising. There appears to be a significant (negative) correlation between these two variables (correlation coefficient is equal to -0.7). Their relation follows a power law with exponent $\alpha = -1.69$, as compared to the linear relation between the average cc of the common venues and the friendship probability (Figure 16). This last result, further supports our above argument that the entropy of a place can be used to tie the two information planes and drive applications such as revealing social affinities from location histories. We will examine the latter in the following section.

6 Revealing Friendships

In the previous section, we have examined the user similarity and various venue-related metrics and their correlation with the users' social relations. To summarize, the feature that appears to be able to capture the best the interplay between the social and spatial components of an LBSN is the cc. However, as explained in Section 5, its strong correlation might be illusive, since its calculation explicitly utilizes the social relationships. We further found that the entropy of the common places visited by two users appears to be correlated with their probability of friendship as well and that friends tend to have higher similarity scores. Our analysis therefore implies that similar metrics can be used to make an educated judgment with regards to the social relationship between two randomly selected users whose spatial behaviors are known in terms of the common venues and their entropy. Alternatively, by utilizing a mix of social and partial spatial location it might be possible to estimate future visits of users. The list of possible applications realized through the bonds between social and spatial information in an LBSN is long and not the focus of our study.

In this section, we want to examine the importance of the metrics we considered for estimating the existing users' affinity relations. In other words, considering the graph in Figure 3 and assuming we are only aware of the solid edges, can we estimate the dashed ones? To reiterate, our goal is not to be able to provide a full fledged clustering algorithm on the affinities/ties of the social graph; we only want to examine the strength of the explored metrics in estimating social relations.

Metric	AUC
# common venues	0.63
User similarity	0.73
Avg entropy	0.71
Min entropy	0.76
Max entropy	0.66
Avg cc	0.88
Min cc	0.65
Max cc	0.94



Metric	Precision	Recall
# common venues	0.11	0.016
User similarity	0.28	0.25
Avg entropy	0.42	0.59
Min entropy	0.44	0.63
Max entropy	0.37	0.46
Avg cc	0.89	0.23
Min cc	0.85	0.3
Max cc	0.85	0.3
Full entropy	0.42	0.63
Full cc	0.83	0.33
All features	0.52	0.6

Fig. 22. The predictive power of each considered feature using simple unsupervised learning algorithms: (i) threshold-based (Left table and figure in the middle) and (ii) k-means (Right table). Full entropy (cc) refers to using all the entropy (cc) related features.

We first consider a simple unsupervised, threshold-based, inference model. In particular, for every pair of users, we compute the following metrics, (i) number of common venues (our baseline), (ii) user similarity, (iii) average/min/max entropy of common venues, and (iv) average/min/max cc of common venues. Then, based on a threshold comparison we classify the pair as being friends or not and we obtain the (fitted) ROC curves for the positive instances presented in Figure 22. We focus on the positive (friends) instances, since there is a strong unbalanced distribution of the friends/non-friends instances in the network. Hence even a simple classifier that states every pair as non-friends, would exhibit a very good overall performance, but it would perform very poorly in the classification of the instances of friends. The table on the left also provides the area under the curve (AUC); the larger the AUC, the better is the quality we have in our assessments (lower false positives and larger true positives). As we expected, the average and min cc provide the best performance, while the entropy metrics together with the user similarity come right after, performing better than the baseline of simply the number of common venues. Establishments that are less diverse in terms of people that socialize there tend to be better indicators of bonds, and this information can be used to accurately infer social relations. Furthermore, balancing the number of common venues between two users with their activity improves the prediction, since it accounts for the user pair's diversity as explained earlier.

We further examine an unsupervised clustering algorithm, that operates on the same set of features. We use a simple k-means algorithm and compute the precision and recall on the positive instances. Briefly, precision is the fraction of friendship predictions that are correct, while recall is the fraction of actual friendships that the algorithm was able to identify. The results are presented in the right table at Figure 22 and as we can see again, the features that consider the type of the venue can significantly improve our inference capability. For example, by using the average entropy of the common venues visited by two people we are able to recover 63% of the actual friendships, while from all our friendship predictions, 44% of them are correct. The corresponding percentages when using only the number of common affiliations, are only 1.6% and 11%; a significant improvement. Our results clearly indicate that metrics such as the entropy and the cc of a venue can help towards the improvement of functionalities such as relationships inference, location prediction etc. It is interesting to observe, that the user similarity performs fairly poor in this test. The reason for this is that there are still friends with low similarity, who are clustered together with the non-friends (low recall), while there are a few non-friends who exhibit larger similarity values and are clustered together with the friends. However, since the friend instances are extremely small in number (as compared to the non-friends), this leads to an overall low precision.

Based on the above results, we believe that defining a user similarity metric that accounts for the entropy of the visited venues can further improve the performance. We seek to examine similar approaches as part of our future work.

7 Conclusions

In this paper we model an LBSN as an affiliation network and by analyzing data from a commercial network we identify bonds between the social and spatial information plane of the system. We find that friends exhibit in general much larger similarity with regards to the number of common venues visited, as compared to non-friends. Considering only the number of common venues between two users, is not very helpful for strongly tying the two components of the network. Even though user similarity can provide a better bonding, the diversity of these common venues with regards to people visiting them is more informative and connect these two parts better. This is also supported by the evaluations and results from simple, unsupervised social link classifiers.

In the future, we seek to examine the location homophily issue in greater detail using longitudinal data and various different similarity metrics (e.g., cosine similarity on an appropriately defined feature vector). As aforementioned, we also opt to intergrate information for the venue entropy in the user similarity and examine any performance improvements in bonding social and spatial information of an LBSN.

References

1. Cheng, Z., Caverlee, J., Lee, K., Sui, D.Z.: Exploring millions of footprints in location sharing services. In: ICWSM (2011)
2. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: Friendship and mobility: User movement in location-based social networks. In: ACM KDD, pp. 279–311 (2011)
3. Christakis, N.A., Fowler, J.H.: The spread of obesity in a large social network over 32 years. New England Journal of Medicine (2007)
4. Cranshaw, J., Toch, E., Hong, J., Kittur, A., Sadeh, N.: Bridging the gap between physical location and online social networks. In: UBICOMP (2010)
5. Eagle, N., Pentland, A.: Eigenbehaviors: identifying structure in routine. In: Behavioral Ecology and Sociobiology (2009)
6. Eagle, N., Pentland, A., Lazer, D.: Inferring friendship network structure by using mobile phone data. Proceedings of the National Academy of Sciences (2009)
7. Friedkin, N.: A structural theory of social influence. Cambridge University Press (1998)
8. Kamdel, D.B.: Homophily, selection and socialization in adolescent friendships. American Journal of Sociology (1978)

9. Lazarsfeld, P.F., Merton, R.K.: Friendship as a social process: A substansive and methodological analysis. In: Freedom and Control in Modern Society (1954)
10. Li, N., Chen, G.: Analysis of a location-based social network. In: IEEE CSE (2009)
11. Li, Q., Zheng, Y., Xie, X., Chen, Y., Liu, W., Ma, W.Y.: Mining user similarity based on location history. In: ACM GIS (2008)
12. Lindqvist, J., Cranshaw, J., Wiese, J., Hong, J., Zimmerman, J.: Im the mayor of my house: Examining why people use foursquare - a social-driven location sharing application. In: ACM CHI (2011)
13. Miklas, A.G., Gollu, K.K., Chan, K.K.W., Saroiu, S., Gummadi, K.P., de Lara, E.: Exploiting Social Interactions in Mobile Systems. In: Krumm, J., Abowd, G.D., Seneviratne, A., Strang, T. (eds.) UbiComp 2007. LNCS, vol. 4717, pp. 409–428. Springer, Heidelberg (2007)
14. Newmann, M.E.J., Watts, D.J., Strogatz, S.H.: Random graph models of social networks. Proceedings of the National Academy of Science (2002)
15. Noulas, A., Scellato, S., Lambiotte, R., Pontil, M., Mascolo, C.: A tale of many cities: universal patters in human urban mobility. PLoS ONE 7(5), e37027 (2011), doi:10.1371/journal.pone.0037027
16. Noulas, A., Scellato, S., Mascolo, C., Pontil, M.: An empirical study of geographic user activity patterns in foursquare. In: ICWSM (poster session) (2011)
17. Noulas, A., Scellato, S., Mascolo, C., Pontil, M.: Exploiting semantic annotations for clustering geographic areas and users in location-based social networks. In: SMW (2011)
18. Scellato, S., Noulas, A., Lambiotte, R., Mascolo, C.: Socio-spatial properties of online location-based social networks. In: ICWSM (2011)
19. Scellato, S., Noulas, A., Mascolo, C.: Exploiting place features in link prediction on location-based social networks. In: ACM KDD (2011)
20. Wang, D., Pedreschi, D., Song, C., Giannotti, F., Barabasi, A.-L.: Human mobility, social ties, and link prediction. In: ACM KDD (2011)
21. Watts, D.J.: Six degrees: The science of a connected age. W.W. Norton & Company (2003)
22. Ye, M., Yin, P., Lee, W.-C.: Location recommendation for location-based social networks. In: ACM GIS (2010)

On Approximation of Real-World Influence Spread

Yu Yang, Enhong Chen, Qi Liu, Biao Xiang, Tong Xu, and Shafqat Ali Shad

Dept. of Computer Science, Univ. of Science and Technology of China,
Hefei 230026, China
{ryanyang,feiniaol,bxiang,tongxu,shafqat}@mail.ustc.edu.cn,
cheneh@ustc.edu.cn

Abstract. To find the most influential nodes for viral marketing, several models have been proposed to describe the influence propagation process. Among them, the *Independent Cascade (IC) Model* is most widely-studied. However, under IC model, computing influence spread (i.e., the expected number of nodes that will be influenced) for each given seed set has been proved to be #P-hard. To that end, in this paper, we propose *GS* algorithm for quick approximation of influence spread by solving a linear system, based on the fact that propagation probabilities in real-world social networks are usually quite small. Furthermore, for better approximation, we study the structural defect problem existing in networks, and correspondingly, propose enhanced algorithms, *GSbyStep* and *SSSbyStep*, by incorporating the *Maximum Influence Path* heuristic. Our algorithms are evaluated by extensive experiments on four social networks. Experimental results show that our algorithms can get better approximations to the IC model than the state-of-the-arts.

1 Introduction

Recently, viral marketing has drawn more and more attention from both industrial and research fields [15]. This kind of marketing is based on the word-of-mouth effect in social networks, i.e., one may be influenced by his neighbors. According to a survey [11], 83% of people prefer consulting family, friends or an expert over traditional advertising before trying a new restaurant, 71% of people do the same before buying a prescription drug, and so on. Thus, companies believe that viral marketing should be one of the most effective marketing strategy.

Unlike traditional marketing strategies, viral marketing only targets a few influential individuals in a social network. For example, if a company employs a viral marketing strategy to promote sales performance, it only needs to choose a small number of individuals and persuade them to be the initial users(by offering them discounted or free products etc.). Due to the word-of-mouth effect, these initial users, also called seed users, will influence their friends to use this product. And once the friends are influenced, they will try to influence their friends and so on. Eventually, a much larger group of people compared to the number of initial users will adopt this product.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 548–564, 2012.

Since different seed users usually lead to different results of word-of-mouth propagation, an important step for a successful viral marketing is to precisely evaluate the expected number of influenced individuals (influence spread) for each seed user set. Along this line, several influence models have been proposed [8][9][12]. Among these models, *Independent Cascade(IC) Model* is a simple and most widely used one that describes the information propagation in a social network as a stochastic process according to certain probabilistic rules [9][12]. However, Wei Chen *et al.* have proved that computing influence spread of a set of seed users, under IC model, is a #P-hard problem [7]. As an alternative, Monte Carlo simulation, which is very time-consuming, is employed to approximately calculate influence spread. For example, for a moderate sized social network, we usually have to run Monte Carlo simulation for more than ten thousands times to obtain a good estimation of the true influence spread. Thus, the problem of efficiently computing influence spread has become a bottleneck for the deployment of viral marketing.

To that end, in this paper, we provide a novel study on approximating influence spread under IC model, mainly based on the observation that the influence propagation probabilities in real-world social networks are usually quite small. Specifically, our main contributions can be summarized as follows:

- To address the above efficiency problem, we show that by solving a linear system we can approximately calculate each node (individual, or user)'s probability of being influenced, and thus we can quickly compute influence spread of a given seed set.
- We point out that *SteadyStateSpread* algorithm [1] is also an approximation for the real influence spread under IC model when the propagation probabilities are small, and this was not illustrated by Aggarwal, *et al.* in [1].
- For better approximation, we discover the structural defect problem (as illustrated by Definition 2 in Section 4.1) existing in networks, and further propose enhanced algorithms by incorporating the Maximum Influence Path(MIP) heuristic [7] to both our algorithm and *SteadyStateSpread*.
- We evaluate our algorithms on four real networks. Experimental results prove our discoveries and show that the proposed algorithms can get good approximations of the real influence spread. Moreover, the nodes ranking obtained by our algorithms are very similar to the true ranking results.

2 Related Work

In general, related work can be grouped into two categories. The first category includes the most relevant work on influence models. In the second category, we introduce the related work on computing social influence spread.

Influence Models. Domingos *et al.* first proposed to mine social networks for marketing [8]. However, their models are essentially descriptive, and prior works [1][7][12][13] following [8] focus more on the models that explicitly represent the step-by-step dynamics of influence.

Among existing influence models, *Independent Cascade(IC) Model* [9][12] is one of the most widely-studied models. In IC model, a social network is represented by a directed graph $G(V, E)$. Each node $v \in V$ denotes an individual in the social network. Each edge (u, v) represents the relationship between u and v, and is assigned with a real number p_{uv} ($p_{uv} \in [0, 1]$) which is the probability that v is influenced by u through the edge in the next step after u is activated. In the beginning, there is a set of users (seed set) who are already activated, i.e. influenced, denoted by S. Let S_t represents the set of nodes that become activated at time t, and $S_0 = S$. At time $t + 1$, each node $u \in S_t$ will try to activate its inactivated neighbor v with probability p_{uv}. If there are no newly activated nodes at time $t+1$, the propagation process ends. The *influence spread* of S, which is denoted by $\sigma(S)$, is the expected number of nodes being activated in the whole propagation process.

Two kinds of IC model, namely *Uniform IC Model* and *Weighted Cascade(WC) Model* are defined in [12]. In uniform IC model, each link shares the same propagation probabilities, while according to WC model, the propagation probability through edge(u,v) equals to $weight(u, v)/indegree(v)$.

Influence Spread Computation. In [12], Kempe, *et al.* exploited running Monte Carlo simulation for a large number of times to evaluate the influence spread of a given seed set under IC model.

Further, Wei Chen *et al.* [7] proved that computing influence spread based on IC model is #P-hard. Given the fact that the influence of a node is always local, they proposed the Maximum Influence Path(MIP) heuristic. Specifically, they defined the *propagation probability* of a path $P =< u = n_1, n_2, ..., n_m = v >$ $(u \in S)$, $pp(P)$, as

$$pp(P) = \prod_{i=1}^{m-1} p_{n_i n_{i+1}}$$

and the maximum influence path from S to v, which is denoted by $MIP(S, v)$, is defined as

$$MIP(S, v) = arg \max_{P} \{pp(P)|P \in Path(S, v)\}$$

in which $Path(S, v)$ is the set of all paths from S to v. By using the MIP heuristic, they regard the probability of v being influenced by seed set S as $pp(MIP(S, v))$. Their experiments showed that by using this heuristic one can accelerate the influence maximization algorithm drastically. However, the influence spread calculated by the MIP heuristic maybe very different with the true influence spread because it abandons too many possible paths.

In contrast, Kimura and Saito proposed the shortest-path based influence models and provided efficient algorithms to compute influence spread under these models in [13]. However, their algorithms are only suitable for the uniform IC model where propagation probabilities of links are all the same, which is seldom the case in reality.

Recently, Aggarwal *et al.* [1] proposed the *SteadyStateSpread* method to compute *flow authority* by solving a system of nonlinear equations,

$$p_v = \begin{cases} 1 & \text{if } v \in S \\ 1 - \prod_{u \in N(v)} (1 - p_{uv} * p_u) & \text{if } v \notin S \end{cases} \tag{1}$$

in which $N(v)$ is the set of all v's in-neighbors and p_v indicates the probability of v being influenced. As we will explain later, Eq.(1) can be used to approximately compute the real influence spread under IC model when propagation probabilities through links are small. However, it does not strictly hold for some situations(e.g., the situation illustrated in Section 4.1). More importantly, there are some difficulties in solving systems of nonlinear equations, such as the convergence problem [6] and the multiple solutions problem [10]. Though Aggarwal *et al.* [1] gave a simple iterative algorithm for solving Eq.(1), they did not theoretically prove that their algorithms can converge or Eq.(1) has only one solution.

3 Approximating Influence Spread by Linear System

In this section, we first illustrate the observation that influence propagation probabilities in real-world social networks are usually quite small. Based on this fact, we then show the way to approximate influence spread, and represent the approximation by a linear system. At last, we propose a simple iterative algorithm to solve the linear system and return the influence spread for each seed set.

3.1 Preliminary Observation

Though there exists the effect of peer-to-peer influence, in real scenarios of information diffusion, propagation probabilities between people are very small. For example, according to [5], in Facebook, the average probability that an individual will share a Web link is about 2% when there are 6 of his friends who shared the same link before. In LiveJournal, the average probability that an individual will join a community is no more than 2% even though 50 of his/her friends have already joined that community [4]. Moreover, influence is not the only reason that leads to phenomena of friends sharing same Web links and joining same communities. Prior works, [2][3][16], showed that correlations between friends in social networks are also an important factor that contributes to homophily phenomena. Thus, the actual propagation probability caused by the effect of influence is even smaller.

To facilitate the following discussion, we first define *small propagation probabilities* mathematically.

Definition 1 (Small Propagation Probabilities). *Given a network $G = (V, E)$, if for $\forall v \in V$, and for $\forall u \in N(v)$, $p_{uv} < weight(u, v)/indegree(v)$, we say the propagation probabilities in G are small. Note that $indegree(v) = \sum_{u \in N(v)} weight(u, v)$.*

From Definition 1 we can see that if the small propagation probabilities condition holds in a social network, then for $\forall v \in V$, $\sum_{u \in N(v)} p_{uv} < 1$ because $p_{uv} < weight(u, v)/indegree(v)$. The further explanations of these two formulas can be easily observed from the real world social networks: First, besides the influence coming from the direct neighbors in the specific social network, each node v's activity is also impacted by the information from other sources (e.g., influence coming from other friends that are not included in the current network). Second, when influence propagates in the network, there is information lost (or decay) in each node, and this has been widely observed and adopted by models from other domains, such as the PageRank method [14].

3.2 Approximation of Real-World Influence Probability

In this subsection we consider approximating influence spread under IC model for social networks, where propagation probabilities are small. We use $\sigma(S)$ to denote influence spread of a seed set S. It is obvious that $\sigma(S) = \sum_v p_v$, in which p_v denotes the probability that v will be influenced, also called the *influence probability* of v. Since $\forall v \in S$, $p_v = 1$, the core step of computing $\sigma(S)$ is to compute p_v for each $v \notin S$. In the following, we show that the influence probability of $v (v \notin S)$ can be well approximated by a linear equation that $p_v = \sum_{u \in N(v)} p_u * p_{uv}$.

Let $v(t)$ denotes the event that v becomes influenced at time t. Note that $p\{v(0)\} = 1$ if $v \in S$. For each node $v \notin S$, $p\{v(t)\}$ can be computed by

$$p\{v(t)\} = \sum_{W \subseteq N(v)} p\{v(t)|W(t-1), \tilde{v}(t-1)\} p\{W(t-1), \tilde{v}(t-1)\} \tag{2}$$

where $N(v)$ is the set of in-neighbors of v, W is a subset of $N(v)$, $W(t-1)$ indicates the event that all nodes in set W become influenced at time $t-1$ and $\tilde{v}(t-1)$ denotes the event that v is still not influenced after time $t-1$. Worth noting that $p\{v(t)\} = p\{v(t), \tilde{v}(t-1)\}$[1]. It is obvious that

$$p\{v(t)|W(t-1), \tilde{v}(t-1)\} = 1 - \prod_{u \in W} (1 - p_{uv}) \tag{3}$$

Following [8] and [13], given the marginals $\{p\{u(t-1)\}; u \in W\}$ and $p\{\tilde{v}(t-1)\}$, we approximate $p\{W(t-1), \tilde{v}(t-1)\}$ by the maximal entropy estimation:

$$p\{W(t-1), \tilde{v}(t-1)\} = p\{\tilde{v}(t-1)\} \prod_{u \in N(v)} p\{u(t-1)\}^{h_u} (1 - p\{u(t-1)\})^{1-h_u} \tag{4}$$

where h_u is an indicator that if $u \in W$, h_u=1, otherwise h_u=0. Combining Eq.(2), Eq.(3) and Eq.(4), and after some algebraic transformations, we have

$$p\{v(t)\} = p\{\tilde{v}(t-1)\}[1 - \prod_{u \in N(v)} (1 - p_{uv} * p\{u(t-1)\})] \tag{5}$$

Note that under the small propagation probabilities condition, p_{uv} is very small, the probability that v will be influenced is even smaller. So $p\{\tilde{v}(t-1)\}$ is very

[1] If v is influenced at time t, it should stay inactive from time 0 to $t-1$.

close to 1 [2]. Thus, $p\{v(t)\}$ can be approximated by

$$p\{v(t)\} = 1 - \prod_{u \in N(v)} (1 - p_{uv} * p\{u(t-1)\}) \tag{6}$$

Since $p_{uv} * p\{u(t-1)\} << 1$ and $(1-a)(1-b) \approx 1 - a - b$ when $a, b << 1$, we can further approximate $p\{v(t)\}$ with a linear equation:

$$p\{v(t)\} = \sum_{u \in N(v)} p_{uv} * p\{u(t-1)\} \tag{7}$$

Now we add up $p\{v(t)\}$ over t to get p_v. Note that $p_v = 1$ if $v \in S$. For $v \notin S$, we have

$$\begin{aligned} p_v &= \sum_{t=0} p\{v(t)\} \\ &= p\{v(0)\} + \sum_{u \in N(v)} p_{uv} \sum_{t=1} p\{u(t-1)\} \\ &= \sum_{u \in N(v)} p_{uv} p_u \end{aligned} \tag{8}$$

Because we are discussing approximation under the condition that $\sum_{u \in N(v)} p_{uv} < 1$, it is guaranteed that $0 \leq p_v < 1$. Thus, the influence probability p_v computed by Eq.(8) is well defined.

Approximation From SteadyStateSpread. As have said, under the small propagation probabilities condition $p_{uv} * p_u << 1$, and recall Eq.(1), it is easy to conclude that $\sum_{u \in N(v)} p_{uv} p_u \approx 1 - \prod_{u \in N(v)} (1 - p_{uv} p_u)$. In this situation, we can see that *SteadyStateSpread* is actually also an approximation for the true influence spread. In the following experiment section, our experimental results verify that the smaller p_{uv} is, the more accurate approximation will be obtained by *SteadyStateSpread.*

3.3 Linear System Formulation

Rewriting Eq.(8) in a matrix form we can get a linear system. To facilitate the following discussion, first we list some math notations in Table 1.

Table 1. Math Notations

Notations	DESCRIPTION
$[\mathbf{M}]_{\vert V \vert * \vert V \vert}$	probability adjacent matrix, $\mathbf{M}_{uv} = p_{uv}$
$\mathbf{P} = [p_1, p_2, ..., p_{\vert V \vert}]^T$	p_v is the probability of v being influenced by seed set S
$\mathbf{B} = [b_1, b_2, ..., b_{\vert V \vert}]^T$	$b_v = 1$ if $v \in S$, otherwise $b_v = 0$
$\mathbf{M}_{\overline{S}}$	matrix which is cut down from matrix $\mathbf{M}$ by removing rows whose index $v \in S$
$\mathbf{M}_{\overline{SS}}$	matrix which is cut down from matrix $\mathbf{M}$ by removing rows and columns whose index $v \in S$

[2] The assumption on this value will be verified in the experimental section 5.3.

For node $v \notin S$, we rewrite Eq.(8) by

$$p_v = \sum_{u \in N(v) \bigwedge u \notin S} p_{uv} * p_u + \sum_{u \in N(v) \bigwedge u \in S} p_{uv} \tag{9}$$

Further, using matrix form to represent the right side of Eq.(9), we have

$$p_v = (\mathbf{M}_{\overline{S}})_v^T \mathbf{P}_{\overline{S}} + (\mathbf{M}^T \mathbf{B})_v \tag{10}$$

Put all nodes v that $v \notin S$ together,

$$\mathbf{P}_{\overline{S}} = (\mathbf{M}_{\overline{SS}})^T \mathbf{P}_{\overline{S}} + (\mathbf{M}^T \mathbf{B})_{\overline{S}} \tag{11}$$

thus we have

$$[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T] \mathbf{P}_{\overline{S}} = (\mathbf{M}^T \mathbf{B})_{\overline{S}} \tag{12}$$

Given that we are dealing with approximations of influence spread under the condition that $\sum_{u \in N(v)} p_{uv} < 1$, the matrix $[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T]$ is strictly diagonally dominant. Thus, $[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T]$ is invertible and the linear system of Eq.(12) has only one solution that $\mathbf{P}_{\overline{S}}$=$[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T]^{-1}(\mathbf{M}^T \mathbf{B})_{\overline{S}}$. In this way, by summing up p_v over each v, we get influence spread for seed set S(i.e., $\sigma(S)$).

3.4 A Simple Iterative Algorithm

As we stated above, to approximately compute the influence spread of a seed set S, we only have to solve the linear equation Eq.(12) and sum up all p_v.

Algorithm 1. GS(G, S)

```
for $v = 1$ to $|V|$ do
    if $v \in S$ then
        $p_v \leftarrow 1$
    else
        $p_v \leftarrow 0$
    end if
end for
while not converge do
    for $v = 1$ to $|V|$ do
        if $v \notin S$ then
            $p_v \leftarrow \sum_{u \in N(v)} p_{uv} * p_u$
        end if
    end for
end while
$sum \leftarrow 0$
for $v = 1$ to $|V|$ do
    $sum \leftarrow sum + p_v$
end for
return $sum$
```

Since $[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T]$ is strictly diagonally dominant, we can use *Gauss-Seidel(GS)* algorithm to efficiently solve Eq.(12) in $O(E)$ time. The entire computation process can be illustrated by Algorithm 1, where the iteration formula used is Eq.(8). Based on the output of Algorithm 1, we can rank each seed set S, so as to find the most influential individuals for marketing.

4 Algorithms Incorporating MIP Heuristic

In this section, we first illustrate the structural defect problem existing in many methods for social networks. Then, to address this problem and to get a better approximation, we propose enhanced algorithms by incorporating the Maximum Influence Path(MIP) heuristic into both *GS* algorithm and *SteadyStateSpread.*

4.1 Structural Defect

According to Eq.(1) and Eq.(8), for $v \notin S$, the value of p_v depends on all of its in-neighbors. However, in fact sometimes a node u's influence probability p_u is independent from some of its in-neighbors. Consider the following example of an undirected network, as shown in Fig.1.

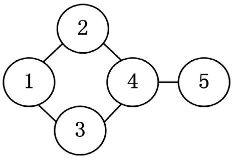


Fig. 1. An undirected network

In this undirected network, assume that node 1 is the only seed node. Based on Eq.(1) and Eq.(8), the probability of node 4 being influenced depends on the probability of node 5 being influenced. However, actually, to influence node 5, node 4 should be influenced first. Thus, p_4 is irrelevant with p_5, and this contradicts with Eq.(1) and Eq.(8). We have reasons to believe that each network, especially an undirected network, may have many sub-structures like Fig.1, and in this situation, if we use Eq.(1) or Eq.(8) to calculate p_v would get misleading results. In summary, we call structures like (node 4, node 5) in Fig.1 *structural defect* of Eq.(1) and Eq.(8), and we define it in a mathematical way.

Definition 2 (Structural Defect). *Given a network $G = (V, E)$, a seed set S and an influence spread algorithm A, if $\exists u, v \notin S$, $\forall P = (p_{n_1}, p_{n_2}, ..., p_{n_m} = v) \in Path(S, v)$, $u \in P$(that is, every path from S to v has to pass u), and according to A, the value of p_u depends on p_v, we say that (u, v) is a* ***structural defect*** *of algorithm A on G.*

4.2 Incorporating Maximum Influence Path Heuristic

To handle the problem of structural defect, we propose to incorporate the Maximum Influence Path(MIP) heuristic [7] into our algorithm. The main idea is to set an iteration threshold $\beta(v)$ for node v, i.e., for each v, we only update p_v in the first $\beta(v)$ iterations.

Let's consider the algorithms of *Gauss-Seidel(GS)* and *SteadyStateSpread* [1], which are both iterative processes of accumulating each path's influence probability for a node (e.g., v) by Eq.(8) and Eq.(1) respectively. Specifically, for a path P from S to v, if we want to include the propagation probability through P, we have to update p_v for no less than $length(P)$(hops of P) iterations. Since the maximum influence path is the path with biggest propagation probability, we do not want to abandon this path's influence. Thus, let $step[v]$ denotes hops of this maximum influence path from S to v, and to include the influence probability of maximum influence path, we should set the iteration threshold for v to be at least $step[v]$.

However, since $step[v]$ is usually also the shortest distance from S to v, and if we just update p_v in the first $step[v]$ rounds of iterations, we may miss some important influences from other paths. For example, the value of p_2 for node 2 in Fig.1 does depend on the influence from all its neighbors. In contrast, if we update p_v for too many iterations, the structural defect will have serious impact on the estimation of influence spread. Take node 4 in Fig.1 for example, as the

Algorithm 2. GSbyStep(G, S)

```
for $v = 1$ to $|V|$ do
    if $v \in S$ then
        $p_v \leftarrow 1$
    else
        $p_v \leftarrow 0$
        calculate hops of the maximum influence path from $S$ to $v$, denoted by $step[v]$
    end if
end for
$times \leftarrow 0$, $updated \leftarrow true$
while $updated$ is $true$ do
    $updated \leftarrow false$, $times \leftarrow times + 1$
    for $v = 1$ to $|V|$ do
        if $v \notin S \bigwedge times \leqslant step[v] + 1$ then
            $p_v \leftarrow \sum_{u \in N(v)} p_{uv} * p_u$
            $updated \leftarrow true$
        end if
    end for
end while
$sum \leftarrow 0$
for $v = 1$ to $|V|$ do
    $sum \leftarrow sum + p_v$
end for
return $sum$
```

round of iterations increases, p_5 will become larger and the structural defect caused by node 5 for p_4 will be more and more serious.

To that end, as a tradeoff, we choose $step[v] + 1$ as v's iteration threshold $\beta(v)$. Choosing such value is because in the $(step[v] + 1)$-th iteration there are few miss counted influence probabilities, and meanwhile, the bad impact from structural defect is limited. Along this line, by incorporating MIP heuristic into Algorithm 1 and *SteadyStateSpread*, we propose *GSbyStep* algorithm(as illustrated by Algorithm 2) and *SSSbyStep* algorithm[3] for better approximations. Worth noting that our solution also works for some other iterative algorithms, where the *structural defect* problem exists.

5 Experimental Evaluation

5.1 Experimental Setup

We evaluate our algorithms by experiments on four real-world datasets (two directed networks and two undirected networks) that we downloaded from SNAP [4]. Detailed information of these four data sets can be seen in Table 2.

Table 2. Description of Data Sets

Name	Description	Type	Nodes	Edges
wiki-Vote	Wikipedia who-votes-on-whom network	Directed	7,115	103,689
p2p-Gnutella04	Gnutella peer to peer network from August 4 2002	Directed	10,876	39,994
email-Enron	Email communication network from Enron	Undirected	36,692	367,662
ca-AstroPh	Collaboration network of Arxiv Astro Physics	Undirected	18,772	396,160

Unlike Kimura and Saito using uniform IC model [13] where propagation probabilities are all the same, for simulating the real-world influence propagation process more accutately, we evaluate our algorithms following more general cases of IC model. Specifically, we set propagation probabilities in a WC model [12] like way. As stated in the related work section, the propagation probability of an edge (u, v) under WC model is $weight(u, v)/indegree(v)$. Thus, this model assumes that the total influence probabilities of a node's in-neighbors are 1. As we have discussed in Section 3.1, this assumption is too idealistic for real scenarios. In order to make the influence propagation process more realistic, and following Definition 1, we slightly change WC model by setting $\sum_{u \in N(v)} p_{uv} = \alpha$, thus $p_{uv} = \alpha * weight(u, v)/indegree(v)$. In this way, we can not only cover more general cases of IC model (by slightly changing weights of the edges) but also capture the characteristics (i.e., small and different values) of the real-world influence propagation probabilities. Thus, in the following, we call such modified model as *General IC Model*, or IC model for short.

[3] Modifying the 14-th line of *GSbyStep* algorithm by replacing $\sum_{u \in N(v)} p_{uv} * p_u$ with $1 - \prod_{u \in N(v)} (1 - p_{uv} * p_u)$, we get the *SSSbyStep* algorithm.

[4] http://snap.stanford.edu

For each network, we conduct three groups of experiment by setting α as 0.5 for all nodes, setting α as 0.75 for all nodes, and randomly generating α with a uniform distribution on [0.1,0.9] for each node, respectively. For evaluation, the average result of 20,000 times Monte Carlo simulation of IC model (*GICM*) is regarded as the real influence spread. In the following, we compare our algorithms (*SSSbyStep*,*GSbyStep*,*GS*) with four baseline algorithms, *MIP* [7], *SteadyStateSpread(SSS)* [1], *SP1M* [13] and an insufficient times Monte Carlo simulation of IC model, i.e., 200 times of Monte Carlo simulation(*GICM200*).

5.2 Ranking Problem

First we consider the problem of extracting influential seeds by ranking candidate nodes. Specifically, we try to find the rank list of $100(|S| = 100)$ influential seeds, i.e., the top-100 nodes with higher influence spread than others.

We use the result of 20,000 times Monte Carlo simulation as the ground truth, and the ranking according to such simulation is regarded as the true nodes rank. For evaluating rankings obtained by different algorithms, we compare similarities between those rankings with the true ranking. We employ the same measurement which is used in [13], the *ranking similarity* $F(k)$. $F(k)$ quantifies the degree of similarity between two ranking methods at rank k, and it is defined as follows: Let $L(k)$ and $L'(k)$ be the respective sets of top-k nodes for the two ranking methods that we compare. Then, $F(k) = |L(k) \bigcap L'(k)| \ / \ k$.

Fig.2-Fig.5 show the experimental results . We can see that rankings obtained by our proposed algorithms and *SteadyStateSpread(SSS)* are very similar

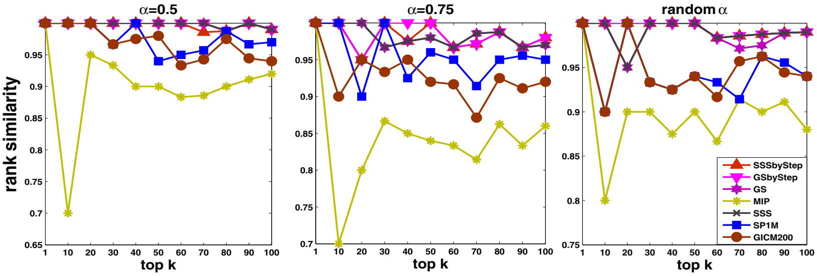


Fig. 2. Rank Similarity Comparison on wiki-Vote Network Data

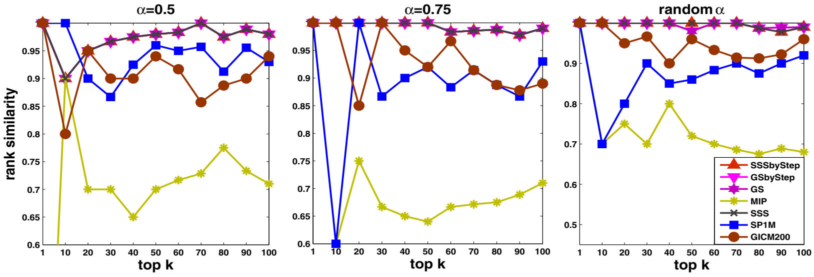


Fig. 3. Rank Similarity Comparison on p2p-Gnutella04 Network Data

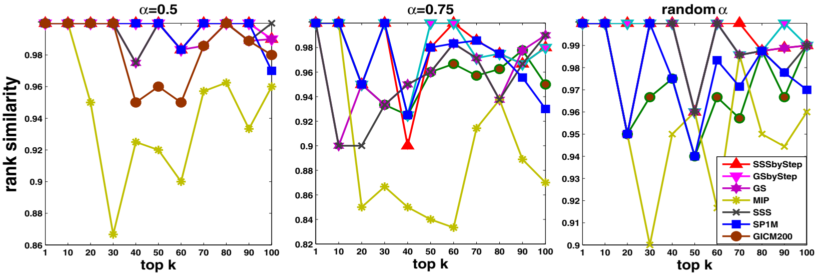


Fig. 4. Rank Similarity Comparison on email-Enron Network Data

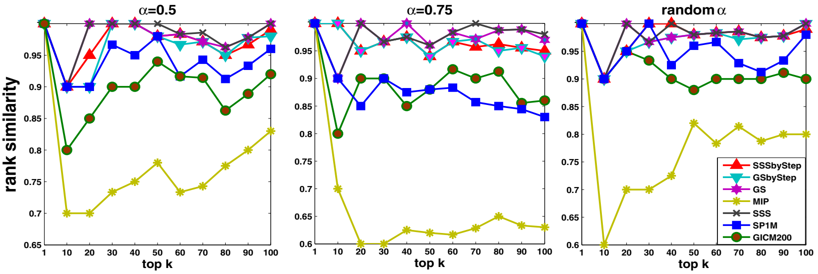


Fig. 5. Rank Similarity Comparison on ca-AstroPh Network Data

to the true ranking while others are not. This is because both our algorithms and *SteadyStateSpread(SSS)* are approximations for the real influence spread as stated in Section 3.

Another observation is that in most cases all algorithms can choose the most influential seed(top-1 node) accurately, even algorithms like *MIP* which is not very effective on ranking similarity metric. This may explain why these algorithms work well for the influence maximization problem [7][12][13]. Since under the greedy framework [12] for solving this problem, an algorithm only cares about finding the node with the maximum increasing influence spread in each round, and this node will be added into seed set.

5.3 Estimation of Influence Spread

We then evaluate each algorithm by their performances on estimating the influence spread for a given seed set. For each dataset, we first calculate influence spreads of the top-1000 nodes with highest out-degrees[5], i.e., experiment with setting the size of seed set equals to 1. Then we randomly generate 100 different seed sets with the size to be 10, 20, 30, 40, 50, respectively. At last, the effectiveness of each algorithm is measured by *error rate* of the influence spread returned by this algorithm. Here, we define $\sigma(S)$ as the true influence spread

[5] Since people often care about top influential nodes, and nodes with higher out-degree are usually more influential.

Table 3. Average Influence Spread ($\sigma(S)$) under $\alpha = 0.5$ Setting

size of seed set dataset	1	10	20	30	40	50
wiki-Vote	4.58	45.22	87.83	133.25	178.60	222.61
p2p-Gnutella04	3.76	37.34	74.15	112.60	148.97	185.13
email-Enron	14.21	151.34	269.06	428.79	559.47	733.68
ca-AstroPh	4.41	44.06	86.04	129.91	171.13	215.14

Table 4. Average Influence Spread ($\sigma(S)$) under $\alpha = 0.75$ Setting

size of seed set dataset	1	10	20	30	40	50
wiki-Vote	8.29	80.76	156.41	234.35	310.25	384.17
p2p-Gnutella04	8.91	89.58	169.30	260.26	343.52	419.03
email-Enron	31.04	318.28	579.39	895.69	1,160.61	1,495.89
ca-AstroPh	10.65	104.97	201.66	301.34	392.89	485.68

Table 5. Average Influence Spread ($\sigma(S)$) under random α Setting

size of seed set dataset	1	10	20	30	40	50
wiki-Vote	4.71	46.34	90.73	136.85	183.16	227.99
p2p-Gnutella04	3.81	37.74	75.29	113.77	151.07	188.41
email-Enron	14.02	150.40	264.93	424.46	552.89	724.63
ca-AstroPh	4.39	44.09	85.69	129.41	170.09	212.86

computed by 20,000 times Monte Carlo simulation, and $\sigma^{'}(S)$ as the influence spread calculated by an approximate algorithm. Then the error rate of $\sigma^{'}(S)$ can be measured by $|\sigma(S) - \sigma^{'}(S)|/\sigma(S)$.

Ground Truth. The average values of $\sigma(S)$ under different α settings are illustrated in Table 3 - Table 5, where a much larger group of nodes are finally influenced compared to the size of each seed set. Based on these tables, we then present a verification for the assumption that $p\{\tilde{v}(t-1)\}$ is very close to 1 for each v and t. For simplicity, we take $\alpha = 0.75$ and the email-Enron network (with the largest number of influenced nodes) as an example. From Table 4 we can see that when $|S| = 50$, the average p_v for each node in email-Enron network is $1,495.89/36,692 = 0.04$, thus $1 - p_v = 0.96$. Since $1 - p_v < p\{\tilde{v}(t-1)\}$, it means $p\{\tilde{v}(t-1)\} > 0.96$. Similarly, when $|S| = 1$, we can get $p\{\tilde{v}(t-1)\} > 0.999$. From these lower bounds we can summarize that $p\{\tilde{v}(t-1)\}$ is very close to 1.

Effectiveness. Correspondingly, Fig.6-Fig.9 show the average error rates of different algorithms when the size of each seed set varies from 1 to 50. From Fig.6 and Fig.7 we can see that our two linear algorithms *GSbyStep* and *GS* are the most accurate algorithms for the two directed networks. Error rates of these two algorithms are lower than 1% in most cases. While from Fig.8 and Fig.9, we find that for the two undirected networks, *SSSbyStep* has the lowest error rate in most cases(with error rate lower than 3%). Worth noting that, for the two undirected networks, when $\alpha = 0.75$, accuracy has been significantly

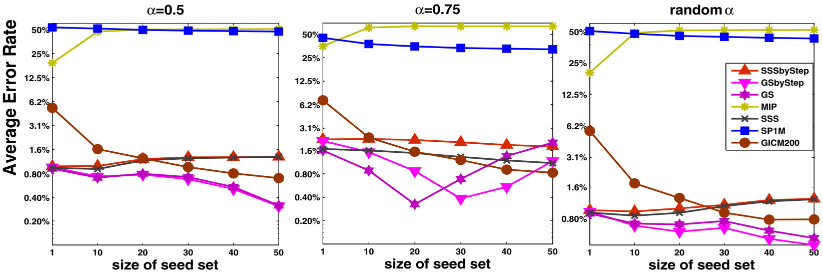


Fig. 6. Average Error Rate Comparison on wiki-Vote Network Data

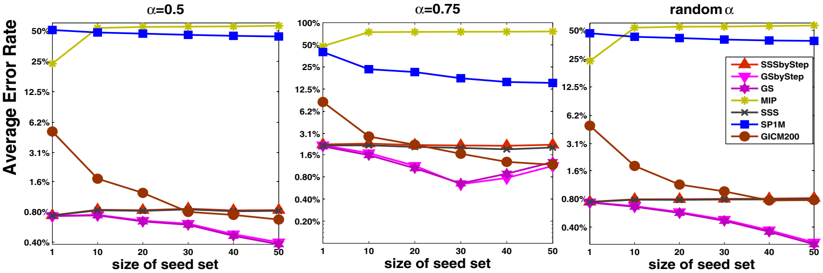


Fig. 7. Average Error Rate Comparison on p2p-Gnutella04 Network Data

improved by incorporating MIP heuristic into algorithms, especially when the size of the seed set is 1. Specifically, average error rate of *SSSbyStep* algorithm is about 7% lower than *SteadyStateSpread(SSS)* algorithm, and average error rate of *GSbyStep* algorithm is about 10% lower than *GS* algorithm.

Another observation is that error rates of our algorithms and *SteadyStateSpread(SSS)* are lower when setting $\alpha = 0.5$ than setting $\alpha = 0.75$. This substantiates that both our algorithms and *SteadyStateSpread* can better approximate the real influence spread when propagation probabilities are smaller.

Efficiency. Our algorithms are also very efficient compared to baseline algorithms. Fig.10 shows the average processing time when $\alpha = 0.75$ and the size of each seed set varies from 1 to 50, where *GICM* denotes the algorithm of 20,000 times Monte Carlo simulation of IC model. Similar results under other α settings are omitted due to the space limit. We can see that tens thousand times Monte Carlo simulation (*GICM*) is very time-consuming. And insufficient times Monte Carlo simulation (*GICM200*) is fast when the size of seed set is small, but the error rate is correspondingly very high. Moreover, another drawback of Monte Carlo simulation is that the processing time increases as seed set gets larger. In contrast, all of our algorithms are iteration-based algorithms, so the running time does not increase when seed sets become larger.

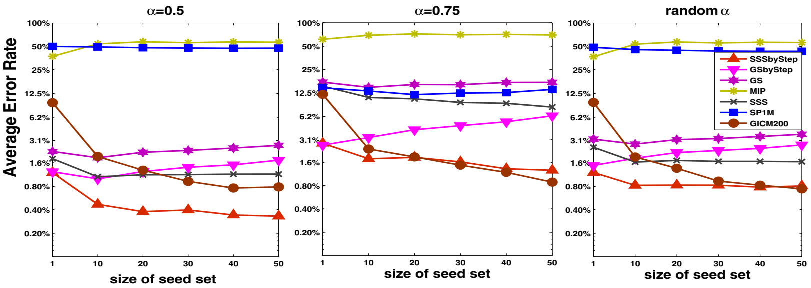


Fig. 8. Average Error Rate Comparison on email-Enron Network Data

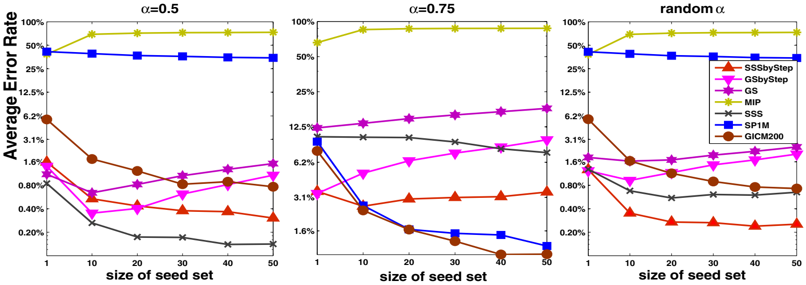


Fig. 9. Average Error Rate Comparison on ca-AstroPh Network Data

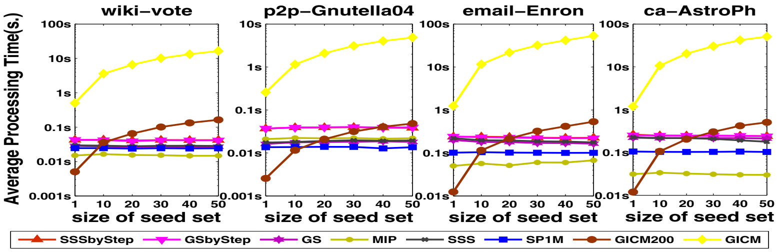


Fig. 10. Average Processing Time($\alpha = 0.75$)

5.4 Effectiveness of Iteration Threshold

In previous experiments, we set $step[v] + 1$ as the iteration threshold for each node v. In this subsection, we provide an experimental proof of this value. We set $\alpha = 0.75$ and use *GSbyStep* algorithm as an example for evaluation, and similar results can be observed for other parameter settings and algorithm *SSSbyStep*. Let $step[v] + r$ be the iteration threshold that we set in *GSbyStep* algorithm for node v. We did 11 groups of experiments of computing influence spreads of the 1000 nodes we picked in Section 5.3 by setting $r = 0, 1, 2, 3..., 10$, respectively. The corresponding average error rates for each r are shown in Fig.11.

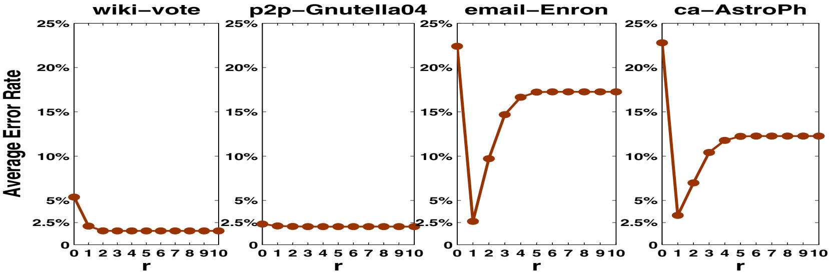


Fig. 11. Average Error Rate under Different Iteration Threshold Settings

From Fig.11 we can see that error rates under different iteration thresholds do not vary very much for the two directed networks, wiki-Vote and p2p-Gnutella04. These results are similar to that in Fig.6 and Fig.7, where the error rates of algorithms incorporating MIP heuristic(*GSbyStep* and *SSSbyStep*) are close to the algorithms not considering such heuristic(*GS* and *SSS*). In contrast, for the two undirected networks, email-Enron and ca-AstroPh, when setting iteration threshold to be $step[v] + 1(r = 1)$ we can get the smallest error rate.

In all, the above results verify that setting iteration threshold to be $step[v]+1$ is reasonable for both directed and undirected networks. In this way, we can not only achieve a better approximation for the real influence spread but also has low computational cost(as shown in Fig.10).

6 Conclusion

In this paper, we exploited the fact that propagation probabilities in real-world social networks are quite small for developing an iterative algorithm *GS* to quickly compute the real influence spread, under IC model. Specifically, we first explain that the influence spread can be well approximated by solving a linear system. Then, we point out the structure defect problem existing in social networks which bothers many iterative computation algorithms. Based on this discovery, we further improve both our *GS* algorithm and the *SteadyStateSpread* algorithm by incorporating the MIP heuristic. Experimental results on four real-world data sets demonstrate that *GS* algorithm can approximate the real influence spread very well. Meanwhile, the improved algorithms(*GSbyStep* and *SSSbyStep*) can achieve better approximations and save computational cost.

Acknowledgments. The work described in this paper was supported by grants from Natural Science Foundation of China (Grant No. 61073110), The National Major Special Science & Technology Projects (Grant No. 2011ZX04016-071), and Research Fund for the Doctoral Program of Higher Education of China (20113402110024), the Key Program of National Natural Science Foundation of China (Grant No. 60933013).

References

1. Aggarwal, C.C., Khan, A., Yan, X.: On Flow Authority Discovery in Social Networks. In: SDM, pp. 522–533 (2011)
2. Anagnostopoulos, A., Kumar, R., Mahdian, M.: Influence and correlation in social networks. In: KDD, pp. 7–15 (2008)
3. Aral, S., Muchnik, L., Sundararajan, A.: Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. Proceedings of the National Academy of Sciences 106(51), 21544–21549 (2009)
4. Backstrom, L., Huttenlocher, D., Kleinberg, J., Lan, X.: Group formation in large social networks: Membership, growth, and evolution. In: KDD, pp. 44–54 (2006)
5. Bakshy, E., Rosenn, I., Marlow, C., Adamic, L.A.: The Role of Social Networks in Information Diffusion. In: WWW, pp. 519–528 (2012)
6. Bus, J.C.P.: Convergence of Newton-Like Methods for Solving Systems of Nonlinear Equations. Numer. Math. 27(3), 271–281 (1977)
7. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: KDD, pp. 1029–1038 (2010)
8. Domingos, P., Richardson, M.: Mining the network value of customers. In: KDD, pp. 57–66 (2001)
9. Goldenberg, K.J., Libai, B., Muller, E.: Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters 12(3), 211–223 (2001)
10. Grosan, C., Abraham, A.: Multiple Solutions for a System of Nonlinear Equations. International Journal of Innovative Computing, Information and Control 4(9), 2161–2170 (2008)
11. Keller, E., Berry, J.: The influentials: One American in ten tells the other nine how to vote, where to eat, and what to buy. The Free Press (2003)
12. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
13. Kimura, M., Saito, K.: Tractable Models for Information Diffusion in Social Networks. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 259–271. Springer, Heidelberg (2006)
14. Langville, A.N., Meyer, C.D.: Deeper Inside PageRank. Internet Mathematics 1(3), 335–380 (2004)
15. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. ACM Transactions on the Web 1(1), 5 (2007)
16. Lewis, K., Gonzalez, M., Kaufman, J.: Social selection and peer influence in an online social network. Proceedings of the National Academy of Sciences 109(1), 68–72 (2012)

Opinion Formation by Voter Model with Temporal Decay Dynamics

Masahiro Kimura[1], Kazumi Saito[2], Kouzou Ohara[3], and Hiroshi Motoda[4]

[1] Department of Electronics and Informatics, Ryukoku University
Otsu 520-2194, Japan
kimura@rins.ryukoku.ac.jp

[2] School of Administration and Informatics, University of Shizuoka
Shizuoka 422-8526, Japan
k-saito@u-shizuoka-ken.ac.jp

[3] Department of Integrated Information Technology, Aoyama Gakuin University
Kanagawa 252-5258, Japan
ohara@it.aoyama.ac.jp

[4] Institute of Scientific and Industrial Research, Osaka University
Osaka 567-0047, Japan
motoda@ar.sanken.osaka-u.ac.jp

Abstract. Social networks play an important role for spreading information and forming opinions. A variety of voter models have been defined that help analyze how people make decisions based on their neighbors' decisions. In these studies, common practice has been to use the latest decisions in opinion formation process. However, people may decide their opinions by taking account not only of their neighbors' latest opinions, but also of their neighbors' past opinions. To incorporate this effect, we enhance the original voter model and define the temporal decay voter (TDV) model incorporating a temporary decay function with parameters, and propose an efficient method of learning these parameters from the observed opinion diffusion data. We further propose an efficient method of selecting the most appropriate decay function from among the candidate functions each with the optimized parameter values. We adopt three functions as the typical candidates: the exponential decay, the power-law decay, and no decay, and evaluate the proposed method (parameter learning and model selection) through extensive experiments. We, first, experimentally demonstrate, by using synthetic data, the effectiveness of the proposed method, and then we analyze the real opinion diffusion data from a Japanese word-of-mouth communication site for cosmetics using three decay functions above, and show that most opinions conform to the TDV model of the power-law decay function.

1 Introduction

Social networking services (SNSs) on the Internet, such as Facebook, Twitter and Digg, have become so popular and use of these services is now a part of our daily activities. Large networks formed by these services play an important role as a medium for spreading diverse information including news, ideas, opinions, and rumors [18,17,8,6]. Users of these services can share their interests or opinions to each other. The resulting social

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 565–580, 2012.

networks and the information propagated therein have great influence on and drastically change our decision making processes and behaviors in daily life. Thus, many attempts have been made to investigate the spread of influence in social networks [15,5,21].

One such typical and well studied problem in social network analysis is the *influence maximization problem*, which is finding a limited number of influential nodes that are effective for spreading information [10,11,16,3,4]. What is common to these studies is that models used allow a node in the network to take only one of the two states, i.e., either active or inactive, because the focus is on *influence*. However, we need a model in which a node can take multiple states for such applications in which a user can choose one from multiple choices. For example, a mobile phone user may change his/her current carrier to the one which the majority of his/her neighbors are using. To model this kind of opinion formation dynamics, a node in the network has to be able to take one of many possible choices as its state. A *voter model* would be the one which is most suitable for this purpose. It is one of the most basic stochastic process models, where a node decision is influenced by its neighbors' decisions [20,9,7,2,22]. We proposed two variants of voter model in our past work: the *value-weighted voter model* that considers opinion values [12], and the *value-weighted mixture voter model* that, in addition to the opinion values, considers the effect of anti-majoritarians, i.e., those people who do not agree with the majority and support the minority opinion [13].

In this paper we also address the problem of opinion formation on the social network, but we especially focus on the fact that our decision may be influenced not only by our neighbors' and our own latest opinions, but also by the neighbors' and our own past opinions. For example, assume that you and your friends have long supported a certain political party, but many of your friends have started changing their supporting party to a different one very recently. Under this situation, you may still stick to your opinion and keep supporting the party, or you may change your mind and follow your neighbors' opinions. This means that your current opinions are influenced not only by the neighbors' latest opinions but also by their past opinions including your own opinions. It is, thus, important to consider all the past opinions in making the current decision. Nonetheless, all the voter models including the two variants mentioned above consider only the latest opinions of its neighbors including itself when updating the opinion of a node.

With this in mind we enhance the original voter model and define the *temporal decay voter (TDV) model* that takes into account all the past opinions discounting the effect of older opinions by using a temporal decay function. The work most closely related to our approach would be the work by Koren [14] which is in the context of recommender systems, where several time drifting user preference models are proposed, some of which adopt a temporal decay function that discounts the effect of older ratings to items. The approach in Koren's work is, unlike our approach, cannot utilize all the past ratings given by a user for an identical item because the user-item matrix that they use does not allow multiple ratings to be stored. In addition, due to the framework of collaborative filtering, it requires the rating history involving multiple items, while our approach can model the temporal dynamics of opinions for a single item. Thus, it does not make sense to compare Koren's approach with ours.

Our major contribution is the following four: 1) the TDV model, 2) an algorithm of learning the parameters of the temporal decay function from the observed opinion spreading data, 3) a model selection method that determines the most appropriate decay function for given data, and 4) new finding regarding the decay model from the analysis of the real data. The model parameters are learned by an efficient iterative algorithm which maximizes the likelihood function. Three representative decay functions are employed, although the framework is not necessarily limited to them: the exponential decay, the power-law decay, and no decay. Which function, each with the optimized parameter values, is most appropriate for given data is determined based on the log likelihood ratio statistic. We evaluate the parameter learning and the model selection methods through extensive experiments using synthetic data with two TDV models, one with the exponential decay and the other with power-low decay. We then applied the methods to the real opinion spreading data from a Japanese word-of-mouth communication site for cosmetics using aforementioned three decay functions, and show that most opinions conform to the TDV model of the power-law decay function.

The paper is organized as follows. We define the TDV model in Section 2 and explain how the model parameters are learned and the most appropriate model is selected in Section 3. The performance of parameter learning and model selection using the synthetic data is reported in Section 4 and the finding from the analysis of real data is reported in Section 5. We end this paper by summarizing the main result in Section 6.

2 Voter Model with Temporal Decay Dynamics

We define the TDV (Temporal Decay Voter) model. Let $G = (V, E)$ be a directed network with self-loops, where V and E ($\subset V \times V$) are the sets of all nodes and links in the network, respectively. Here, $(u, v) \in E$ denotes a (directed) link from node u to node v. When there is a link (u, v), we assume that v can be influenced by its neighbor u in opinion formation process. For a node $v \in V$, let $B(v)$ denote the set of neighbors of v in G, that is,

$$B(v) = \{u \in V;\ (u, v) \in E\}.$$

Note that $v \in B(v)$. Given an integer K with $K \geq 2$, we consider the spread of K opinions (opinion $1, \cdots$, opinion K) on G, where each node holds exactly one of the K opinions at any time t (≥ 0). We assume that each node of G initially holds one of the K opinions with equal probability at time $t = 0$. We denote by

$$g_t : V \rightarrow \{1, \cdots, K\}$$

the *opinion distribution* at time t, where $g_t(v)$ stands for the opinion of node v at time t. Note that g_0 stands for the initial opinion distribution. For any $v \in V$ and $k \in \{1, 2, \cdots, K\}$, let $U_k(t, v)$ be the set of v's neighbors that hold opinion k as its latest opinion (before time t), i.e.,

$$U_k(t, v) = \{u \in B(v);\ \varphi_t(u) = k\},$$

where $\varphi_t(u)$ is the latest opinion of u (before time t).

2.1 Voter Model

We first recall the definition of the voter model (see, e.g., [13]), which is one of the standard models of opinion dynamics, where K is usually set to 2. The evolution process of the voter model is defined as follows:

1. At time 0, each node v independently decides its update time t according to some probability distribution such as an exponential distribution with parameter $\gamma_v = 1$.[1] The successive update time is determined similarly at each update time t.
2. At an update time t, the node v adopts the opinion of a randomly chosen neighbor u, i.e.,
$$g_t(v) \;=\; \varphi_t(u).$$
3. The process is repeated from the initial time $t = 0$ until the next update-time passes a given final-time T_1.

We note that in the voter model each individual tends to adopt the majority opinion among its neighbors.[2] Here note that the definition of one's neighbors include oneself because of the existence of self loop. Thus, we can extend the original voter model with 2 opinions to a voter model with K opinions by replacing Step 2 with: At an update time t, the node v selects one of the K opinions according to the probability distribution,

$$P(g_t(v) = k) \;=\; \frac{|U_k(t, v)|}{|B(v)|}, \quad (k = 1, \cdots, K). \tag{1}$$

2.2 Temporal Decay Voter Model

As mentioned earlier, people may decide their opinions by taking account not only of their neighbors' latest opinions, but also of their neighbors' past opinions including their own opinions. In order to model this kind of situation, for any $t > 0$ and $v \in V$, we consider the set $M(t, v)$ consisting of the time τ ($< t$) at which an individual (a node) v manifested his/her opinion. For $k = 1, \cdots, K$, we also consider a subset of $M(t, v)$,

$$M_k(t, v) \;=\; \{\tau \in M(t, v);\; g_\tau(v) = k\},$$

where $M_k(t, v)$ is the set of node v's opinion manifestation time instances before time t in which v takes opinion k. Now, we can define a voter model which takes all the past opinions into consideration. In this model, Eq. (1) is replaced with

$$P(g_t(v) = k) \;=\; \frac{1 + \sum_{u \in B(v)} |M_k(t, u)|}{K + \sum_{u \in B(v)} |M(t, u)|}, \quad (k = 1, \cdots, K), \tag{2}$$

where we employed a Bayesian prior known as the Laplace smoothing. Here we note that the Laplace smoothing of Eq. (2) corresponds to the assumption that each node initially holds one of the K opinions with equal probability at time $t = 0$. Note also that the

[1] This assumes that the average delay time is 1.

[2] In reality there may be a case that one changes its opinion to a medium one (say 3) listening to two opposite opininons (say 1 and 5). The voter model does not consider this possibility unless at least one of the neighbors has already the medium opinion (3).

Laplace smoothing corresponds to a special case of Dirichlet distributions that are very often used as prior distributions in Bayesian statistics, and in fact the Dirichlet distribution is the conjugate prior of the categorical distribution and multinomial distribution. We refer to this voter model as the *base TDV model.*

Thus far, we assumed that all the past opinions are equally weighted. However, it is naturally conceivable that the quite old opinions have almost no influence. Older opinions are less influential in general. In order to reflect this kind of effects into the model, we consider introducing some decay functions. The simplest one is an exponential decay function defined by

$$\rho(\Delta t; \lambda) \;=\; \exp(-\lambda \Delta t), \tag{3}$$

where $\lambda \geq 0$ is a parameter and $\Delta t = t - \tau$ stands for the time difference between the opinion adoption time t and the opinion manifestation time τ. Another natural one would be a power-law decay function defined by

$$\rho(\Delta t; \lambda) \;=\; (\Delta t)^{-\lambda} \;=\; \exp(-\lambda \log \Delta t), \tag{4}$$

where $\lambda \geq 0$ is a parameter.

Now, we construct a more general decay function. For a given positive integer J, let $f_1(\Delta t), \cdots, f_J(\Delta t)$ be functions on $(0, +\infty)$ such that 1, $f_1(\Delta t)$, $\cdots$, $f_J(\Delta t)$ are linearly independent, that is, if $\lambda_0, \lambda_1, \cdots, \lambda_J$ are real numbers and satisfy

$$\lambda_0 + \sum_{j=1}^{J} \lambda_j f_j(\Delta t) \;=\; 0, \quad (\forall \Delta t \in (0, +\infty)),$$

then $\lambda_0 = \lambda_1 = \cdots = \lambda_J = 0$. We then consider a J-dimensional feature vector,

$$\boldsymbol{F}_J(\Delta t) \;=\; (f_1(\Delta t), \cdots, f_J(\Delta t))^T,$$

where $\boldsymbol{a}^T$ denote the transpose of column vector $\boldsymbol{a}$. For a J-dimensional real column vector with non-negative elements,

$$\boldsymbol{\lambda}_J \;=\; (\lambda_1, \cdots, \lambda_J)^T,$$

which is a parameter vector, we define a decay function $\rho(\Delta t; \boldsymbol{\lambda}_J)$ by

$$\rho(\Delta t; \boldsymbol{\lambda}_J) \;=\; \exp\left(-\boldsymbol{\lambda}_J{}^T \boldsymbol{F}_J(\Delta t)\right), \tag{5}$$

where the matrix operations are used. Representative candidates of feature vector $\boldsymbol{F}_J(\Delta t)$ include

$$\boldsymbol{F}_1(\Delta t) = \Delta t, \quad \boldsymbol{F}_1(\Delta t) = \log \Delta t, \quad \boldsymbol{F}_1(\Delta t) = (\Delta t)^2$$

for $J = 1$,

$$\boldsymbol{F}_2(\Delta t) = (\Delta t,\, \log \Delta t)^T, \quad \boldsymbol{F}_2(\Delta t) = \left(\Delta t,\, (\Delta t)^2\right)^T, \quad \boldsymbol{F}_2(\Delta t) = \left(\log \Delta t,\, (\Delta t)^2\right)^T$$

for $J = 2$,

$$\boldsymbol{F}_3(\Delta t) \;=\; \left(\Delta t,\, \log \Delta t,\, (\Delta t)^2\right)^T$$

for $J = 3$, etc. Note that $\rho(\Delta t; \boldsymbol{\lambda}_J)$ becomes the exponential decay function if $J = 1$ and $\boldsymbol{F}_J(\Delta t) = \Delta t$, and the power-law decay function if $J = 1$ and $\boldsymbol{F}_J(\Delta t) = \log \Delta t$.

Using our general decay function $\rho(\Delta t; \boldsymbol{\lambda}_J)$ (see Eq. (5)), we define the TDV (Temporal Decay Voter) model in the following way. In this model, Eq. (1) is replaced with

$$P(g_t(v) = k) \;=\; \frac{1 + \sum_{u \in B(v)} \sum_{\tau \in M_k(t,u)} \rho(t - \tau; \boldsymbol{\lambda}_J)}{K + \sum_{u \in B(v)} \sum_{\tau \in M(t,u)} \rho(t - \tau; \boldsymbol{\lambda}_J)}, \quad (k = 1, \cdots, K). \tag{6}$$

Here note that Eq. (6) is reduced to Eq. (2) when $\boldsymbol{\lambda}_J$ is the J-dimensional zero-vector $\mathbf{0}_J$, that is, the TDV model of $\boldsymbol{\lambda}_J = \mathbf{0}_J$ coincides with the base TDV model.

3 Learning Method

We consider the problem of identifying the TDV model on network G from an observed data $\mathcal{D}_{T_0}$ in time-span $[0, T_0]$, where $\mathcal{D}_{T_0}$ consists of a sequence of (k, t, v) such that node v changed its opinion to opinion k at time t for $0 \leq t \leq T_0$. The identified model can be used to predict how much of the share each opinion will have at a future time T_1 $(> T_0)$, and to identify both high decay tendency data sets and low decay tendency data sets.

3.1 Parameter Estimation

We describe a method for estimating decay parameter values of the TDV model from a given observed opinion spreading data $\mathcal{D}_{T_0}$. Based on the evolution process of our model (see Eq. (6)), we can obtain the likelihood function,

$$\mathcal{L}(\mathcal{D}_{T_0}; \boldsymbol{\lambda}_J) \;=\; \log\left(\prod_{(k,t,v) \in \mathcal{D}_{T_0}} P(g_t(v) = k) \right), \tag{7}$$

where $\boldsymbol{\lambda}_J$ stands for the J-dimensional vector of decay parameter values, as explained in the previous subsection. Thus our estimation problem is formulated as a maximization problem of the objective function $\mathcal{L}(\mathcal{D}_{T_0}; \boldsymbol{\lambda}_J)$ with respect to $\boldsymbol{\lambda}_J$.

We derive an iterative algorithm for obtaining the maximum likelihood estimators. From the definitions of $P(g_t(v) = k)$ (see Eq. (6)) and $\rho(\Delta t; \boldsymbol{\lambda}_J)$ (see Eq. (5)), we can express Eq. (7) as follows:

$$\begin{aligned} \mathcal{L}(\mathcal{D}_{T_0}; \boldsymbol{\lambda}_J) = & \sum_{(k,t,v) \in \mathcal{D}_{T_0}} \log\left(1 + \sum_{u \in B(v)} \sum_{\tau \in M_k(t,u)} \exp\left(-{\boldsymbol{\lambda}_J}^T \boldsymbol{F}_J(t - \tau) \right) \right) \\ & - \sum_{(k,t,v) \in \mathcal{D}_{T_0}} \log\left(K + \sum_{u \in B(v)} \sum_{\tau \in M(t,u)} \exp\left(-{\boldsymbol{\lambda}_J}^T \boldsymbol{F}_J(t - \tau) \right) \right). \end{aligned} \tag{8}$$

Now, let $\bar{\boldsymbol{\lambda}}_J$ be the current estimate of $\boldsymbol{\lambda}_J$. We foucus on the first term of the right-hand side of Eq. (8), and define $q_{k,t,v}(\tau; \boldsymbol{\lambda}_J)$ by

$$q_{k,t,v}(\tau; \boldsymbol{\lambda}_J) \;=\; \frac{\exp\left(-{\boldsymbol{\lambda}_J}^T \boldsymbol{F}_J(t - \tau) \right)}{1 + \sum_{u \in B(v)} \sum_{\tau' \in M_k(t,u)} \exp\left(-{\boldsymbol{\lambda}_J}^T \boldsymbol{F}_J(t - \tau') \right)}$$

for any $k \in \{1, \cdots, K\}$, $t \in (0, T]$, $v \in V$, and $\tau \in \bigcup_{u \in B(v)} M_k(t, u)$. Note that for any $(k, t, v) \in \mathcal{D}_{T_0}$,

$$q_{k,t,v}(\tau; \boldsymbol{\lambda}_J) > 0, \quad \left(\forall \tau \in \bigcup_{u \in B(v)} M_k(t, u) \right), \tag{9}$$

$$\sum_{u \in B(v)} \sum_{\tau \in M_k(t,u)} q_{k,t,u}(\tau; \boldsymbol{\lambda}_J) + \frac{1}{1 + \sum_{u \in B(v)} \sum_{\tau' \in M_k(t,u)} \exp\left(-\boldsymbol{\lambda}_J{}^T \boldsymbol{F}_J(t - \tau')\right)} = 1. \tag{10}$$

We can transform our objective function as follows:

$$\mathcal{L}(\mathcal{D}_{T_0}; \boldsymbol{\lambda}_J) = \mathcal{Q}\left(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J\right) - \mathcal{H}\left(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J\right), \tag{11}$$

where $\mathcal{Q}\left(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J\right)$ is defined by

$$\begin{aligned} \mathcal{Q}\left(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J\right) = &- \sum_{(k,t,v) \in \mathcal{D}_{T_0}} \sum_{u \in B(v)} \sum_{\tau \in M_k(t,u)} q_{k,t,v}\left(\tau; \overline{\boldsymbol{\lambda}}_J\right) \boldsymbol{\lambda}_J{}^T \boldsymbol{F}_J(t - \tau) \\ &- \sum_{(k,t,v) \in \mathcal{D}_{T_0}} \log\left(K + \sum_{u \in B(v)} \sum_{\tau \in M(t,u)} \exp\left(-\boldsymbol{\lambda}_J{}^T \boldsymbol{F}_J(t - \tau)\right) \right), \end{aligned} \tag{12}$$

and $\mathcal{H}\left(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J\right)$ is defined by

$$\begin{aligned} \mathcal{H}\left(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J\right) = \sum_{(k,t,v) \in \mathcal{D}_{T_0}} &\left\{ \sum_{u \in B(v)} \sum_{\tau \in M_k(t,u)} q_{k,t,v}\left(\tau; \overline{\boldsymbol{\lambda}}_J\right) \log q_{k,t,v}(\tau; \boldsymbol{\lambda}_J) \right. \\ &+ \frac{1}{1 + \sum_{u \in B(v)} \sum_{\tau' \in M_k(t,u)} \exp\left(-\overline{\boldsymbol{\lambda}}_J{}^T \boldsymbol{F}_J(t - \tau')\right)} \\ &\left. \times \log\left(\frac{1}{1 + \sum_{u \in B(v)} \sum_{\tau' \in M_k(t,u)} \exp\left(-\boldsymbol{\lambda}_J{}^T \boldsymbol{F}_J(t - \tau')\right)} \right) \right\}. \end{aligned} \tag{13}$$

By Eqs. (9), (10), (13), and the property of the KL-divergence, it turns out that $\mathcal{H}\left(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J\right)$ is maximized at $\boldsymbol{\lambda}_J = \overline{\boldsymbol{\lambda}}_J$. Hence, we can increase the value of $\mathcal{L}(\mathcal{D}_{T_0}; \boldsymbol{\lambda}_J)$ by maximizing $\mathcal{Q}\left(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J\right)$ with respect to $\boldsymbol{\lambda}_J$ (see Eq. (11)).

We derive an update formula for maximizing $\mathcal{Q}(\boldsymbol{\lambda}_J; \overline{\boldsymbol{\lambda}}_J)$. We foucus on the second term of the right-hand side of Eq. (12) (see also the second term of the right-hand side of Eq. (8)), and define $r_{t,v}(\tau; \boldsymbol{\lambda}_J)$ by

$$r_{t,v}(\tau; \boldsymbol{\lambda}_J) = \frac{\exp\left(-\boldsymbol{\lambda}_J{}^T \boldsymbol{F}_J(t - \tau)\right)}{K + \sum_{u \in B(v)} \sum_{\tau' \in M(t,u)} \exp\left(-\boldsymbol{\lambda}_J{}^T \boldsymbol{F}_J(t - \tau')\right)} \tag{14}$$

for any $t \in (0, T]$, $v \in V$, and $\tau \in \bigcup_{u \in B(v)} M(t, u)$. Note that for any $(k, t, v) \in \mathcal{D}_{T_0}$,

$$r_{t,v}(\tau; \boldsymbol{\lambda}_J) > 0, \quad \left(\forall \tau \in \bigcup_{u \in B(v)} M(t, u) \right), \qquad \sum_{u \in B(v)} \sum_{\tau \in M(t,u)} r_{t,u}(\tau; \boldsymbol{\lambda}_J) < 1. \tag{15}$$

From Eqs. (12) and (14), we can easily see that the gradient vector of $Q\left(\boldsymbol{\lambda}_J;\bar{\boldsymbol{\lambda}}_J\right)$ with respect to $\boldsymbol{\lambda}_J$ is given by

$$\frac{\partial Q\left(\boldsymbol{\lambda}_J;\bar{\boldsymbol{\lambda}}_J\right)}{\partial \boldsymbol{\lambda}_J} = -\sum_{(t,v,k)\in\mathcal{D}_{T_0}}\sum_{u\in B(v)}\left(\sum_{\tau\in M_k(t,u)} q_{t,v,k}\left(\tau;\bar{\boldsymbol{\lambda}}_J\right)\boldsymbol{F}_J(t-\tau) - \sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\boldsymbol{F}_J(t-\tau)\right). \tag{16}$$

Moreover, from Eqs. (14) and (16), we can obtain the Hessian matrix of $Q\left(\boldsymbol{\lambda}_J;\bar{\boldsymbol{\lambda}}_J\right)$ as follows:

$$\frac{\partial^2 Q\left(\boldsymbol{\lambda}_J;\bar{\boldsymbol{\lambda}}_J\right)}{\partial \boldsymbol{\lambda}_J \partial {\boldsymbol{\lambda}_J}^T} = -\sum_{(k,t,v)\in\mathcal{D}_{T_0}}\left\{\sum_{u\in B(v)}\sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\,\boldsymbol{F}_J(t-\tau)\,\boldsymbol{F}_J(t-\tau)^T - \left(\sum_{u\in B(v)}\sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\,\boldsymbol{F}_J(t-\tau)\right)\left(\sum_{u\in B(v)}\sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\,\boldsymbol{F}_J(t-\tau)\right)^T\right\}. \tag{17}$$

By Eq. (17), for any J-dimensional real column vector $\boldsymbol{x}_J$, we have

$$\begin{aligned}
&{\boldsymbol{x}_J}^T\,\frac{\partial^2 Q\left(\boldsymbol{\lambda}_J;\bar{\boldsymbol{\lambda}}_J\right)}{\partial \boldsymbol{\lambda}_J \partial {\boldsymbol{\lambda}_J}^T}\,\boldsymbol{x}_J \\
&= -\sum_{(k,t,v)\in\mathcal{D}_{T_0}}\left\{\sum_{u\in B(v)}\sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\left({\boldsymbol{x}_J}^T\,\boldsymbol{F}_J(t-\tau)\right)^2 - \left(\sum_{u\in B(v)}\sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\,{\boldsymbol{x}_J}^T\,\boldsymbol{F}_J(t-\tau)\right)^2\right\} \\
&= -\sum_{(k,t,v)\in\mathcal{D}_{T_0}}\left\{\sum_{u\in B(v)}\sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\left({\boldsymbol{x}_J}^T\,\boldsymbol{F}_J(t-\tau) - \sum_{u\in B(v)}\sum_{\tau'\in M(t,u)} r_{t,v}(\tau';\boldsymbol{\lambda}_J)\,{\boldsymbol{x}_J}^T\,\boldsymbol{F}_J(t-\tau')\right)^2\right. \\
&\qquad\left. + \left(1 - \sum_{u\in B(v)}\sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\right)\left(\sum_{u\in B(v)}\sum_{\tau\in M(t,u)} r_{t,v}(\tau;\boldsymbol{\lambda}_J)\,{\boldsymbol{x}_J}^T\,\boldsymbol{F}_J(t-\tau)\right)^2\right\}.
\end{aligned}$$

Thus, by Eq. (15), we obtain

$${\boldsymbol{x}_J}^T\,\frac{\partial^2 Q\left(\boldsymbol{\lambda}_J;\bar{\boldsymbol{\lambda}}_J\right)}{\partial \boldsymbol{\lambda}_J \partial {\boldsymbol{\lambda}_J}^T}\,\boldsymbol{x}_J \;\leq\; 0, \qquad \left(\forall \boldsymbol{x}_J \in \mathbf{R}^J\right),$$

that is, the Hessian matrix is negative semi-definite. Hence, by solving the equation

$$\frac{\partial Q\left(\lambda_J; \overline{\lambda}_J\right)}{\partial \lambda_J} = \mathbf{0}_J$$

(see Eq. (16)), we can find the value of λ_J that maximizes $Q\left(\lambda_J; \overline{\lambda}_J\right)$. We employed a standard Newton Method in our experiments.

3.2 Model Selection

One of the important purposes of introducing the TDV model is to analyze how people are affected by their neighbors' past opinions for a specific opinion formation process. In what follows, for a given set of candidate decay functions (i.e., feature vectors), we consider selecting one being the most appropriate to the observed data $\mathcal{D}_{T_0}$ of $|\mathcal{D}_{T_0}| = N$, where N represents the number of opinion manifestations by individuals.

As mentioned in Section 2, the base TDV model is a special TDV model equipped with the decay function that equally weights all the past opinions. Thus, we first examine whether or not the TDV model equipped with a candidate decay function can be more appropriate to the observed data $\mathcal{D}_{T_0}$ than the base TDV model.[3] To this end, we employ the likelihood ratio test. For a given feature vector $\boldsymbol{F}_J(\varDelta t)$, let $\hat{\lambda}_J(\boldsymbol{F}_J)$ be the maximal likelihood estimator of the TDV model equipped with the decay function of $\boldsymbol{F}_J(\varDelta t)$. Since the base TDV model is the TDV model of $\lambda_J = \mathbf{0}_J$, the log-likelihood ratio statistic of the TDV model with $\boldsymbol{F}_J(\varDelta t)$ against the base TDV model is given by

$$Y_N(\boldsymbol{F}_J) = \mathcal{L}\left(\mathcal{D}_{T_0}; \hat{\lambda}_J(\boldsymbol{F}_J)\right) - \mathcal{L}\left(\mathcal{D}_{T_0}; \mathbf{0}_J\right). \tag{18}$$

It is well known that $2Y_N(\boldsymbol{F}_J)$ asymptotically approaches to the χ^2 distribution with J degrees of freedom as N increases. We set a significance level α $(0 < \alpha < 1)$, say $\alpha = 0.005$, and evaluate whether or not the TDV model with $\boldsymbol{F}_J(\varDelta t)$ fits significantly better than the base TDV model by comparing $2Y_N(\boldsymbol{F}_J)$ to $\chi_{J,\alpha}$. Here, $\chi_{J,\alpha}$ denotes the upper α point of the χ^2 distribution of J degrees of freedom, that is, it is the positive number z such that

$$\frac{1}{\Gamma(J/2)2^{J/2}} \int_0^z y^{J/2-1} \exp\left(-\frac{y}{2}\right) dy = 1 - \alpha,$$

where $\Gamma(s)$ is the gamma function. We consider the set $\mathcal{FV}$ of the candidate feature vectors (i.e., decay functions) selected by this likelihood ratio test at significance level α. Next, we find the feature vector $\boldsymbol{F}^*_{J^*}(\varDelta t) \in \mathcal{FV}$ such that it maximizes the log-likelihood ratio statistic $Y_N(\boldsymbol{F}_J)$, $(\boldsymbol{F}_J(\varDelta t) \in \mathcal{FV})$, (see Eq. (18)), and propose selecting the TDV model equipped with the decay function of $\boldsymbol{F}^*_{J^*}(\varDelta t)$. If the set $\mathcal{FV}$ is empty, we select the base TDV model for $\mathcal{D}_{T_0}$.

[3] The base TDV model is not the only baseline model with which the proposed method is to be compared. The simplest one would be the random opininon model in which each user chooses its opinionn randomly independent of its neighbors.

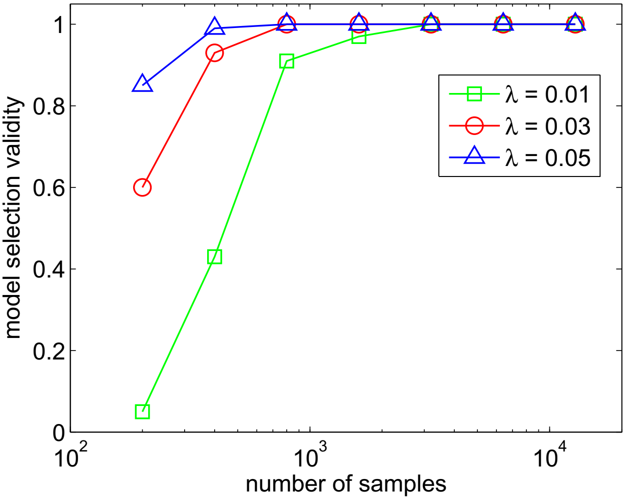


Fig. 1. Results of model selection validity for the exponential TDV model

Here we recall that typical decay functions in natural and social sciences include the exponential decay function (see Eq. (3)) and the power-law decay functions (see Eq. (4)). We refer to the TDV models of the exponential and the power-law decay functions as the *exponential TDV model* and the *power-law TDV model*, respectively. In our experiments, we in particular focus on investigating which of the base, the exponential, and the power-law TDV models best fits to the observed data $\mathcal{D}_{T_0}$. Thus, the TDV model to be considered has $J = 1$ and parameter λ.

4 Evaluation by Synthetic Data

Using synthetic data, we examined the effectiveness of the proposed method for parameter estimation and model selection. We assumed complete networks for simplicity. According to the TDV model, we artificially generated an opinion diffusion sequence $\mathcal{D}_{T_0}$ consisting of 3-tuple (k, t, v) of opinion k, time t and node v such that $|\mathcal{D}_{T_0}| = N$, and applied the proposed method to the observed data $\mathcal{D}_{T_0}$, where the significance level $\alpha = 0.005$ was used for model selection. As mentioned in the previous section, we assumed two cases where the true decay follows the exponential distribution (see Eq. (3)) and the power-law distribution (see Eq. (3)), respectively. Let Y_N^e and Y_N^p denote the log-likelihood ratio statistics of the exponential and the power-law TDV models against the base TDV model, respectively (see Eq. (18)). We varied the value of parameter λ in the following range: $\lambda = 0.01, 0.03, 0.05$ for the exponential TDV model, and $\lambda = 0.4, 0.5, 0.6$ for the power-law TDV model, on the basis of the analysis performed for the real world @cosme dataset (see, Section 5). We conducted 100 trials varying the observed data $\mathcal{D}_{T_0}$ of $|\mathcal{D}_{T_0}| = N$, and evaluated the proposed method.

First, we investigated the model selection validity $\mathcal{F}_N/100$, where $\mathcal{F}_N$ is the number of trials in which the true model was successfully selected by the proposed method.

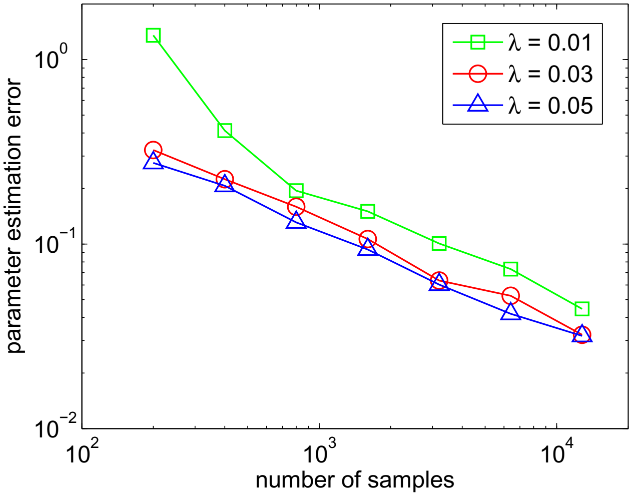


Fig. 2. Results of Parameter estimation error for the exponential TDV model

Namely, if the exponential TDV model is the true model, then $\mathcal{F}_N$ is defined by the number of trials such that

$$2Y_N^e > \max\left(\chi_{1,\alpha}, 2Y_N^p\right),$$

and if the power-law TDV model is the true model, then $\mathcal{F}_N$ is defined by the number of trials such that

$$2Y_N^p > \max\left(\chi_{1,\alpha}, 2Y_N^e\right).$$

Second, we examined the parameter estimation error $\mathcal{E}_N$ for the trials in which the true model was selected by the proposed method. Here, $\mathcal{E}_N$ is defined by

$$\mathcal{E}_N = \frac{|\hat{\lambda}(N) - \lambda^*|}{\lambda^*},$$

where λ^* is the true value of parameter λ, and $\hat{\lambda}(N)$ is the value estimated by the proposed method from the observed data $\mathcal{D}_{T_0}$ of $|\mathcal{D}_{T_0}| = N$. Figures 1 and 2 show the results for the exponential TDV model, and Figures 3 and 4 show the results for the power-law TDV model. Here, Figures 1 and 3 display model selection validity $\mathcal{F}_N/100$ as a function of sample size N. Figures 2 and 4 display parameter estimation error $\mathcal{E}_N$ as a function of sample size N. As expected, $\mathcal{F}_N$ increases and $\mathcal{E}_N$ decreases as N increases. Moreover, as λ becomes larger, $\mathcal{F}_N$ increases and $\mathcal{E}_N$ decreases. Note that a large λ means quickly forgetting past activities, and a small λ means slowly forgetting them. Thus, we can consider that a TDV model of smaller λ requires more samples to correctly learn the model. From Figures 1, 2, 3 and 4, we observe that the proposed method can work almost perfectly when N is greater than 500, and λ is greater than 0.01 for the exponential TDV model and greater than 0.4 for the power-law TDV model.

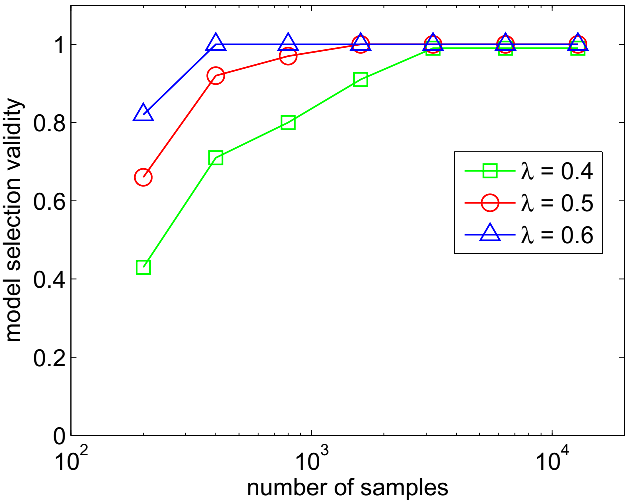


Fig. 3. Results of model selection validity for the power-law TDV model

5 Findings in Opinion Formation on Social Media

5.1 Dataset

We collected real data from "@cosme"[4], which is a Japanese word-of-mouth communication website for cosmetics. In @cosme, a user can post a review and give a score of each brand (one from 1 to 7). When one user registers another user as his/her favorite user, a "fan-link" is created between them. We traced up to ten steps in the fan-links from a randomly chosen user in December 2009, and collected a set of (b, k, t, v)'s, where (b, k, t, v) means that user v scored brand b k points at time t. The number of brands was 7,139, the number of users was 45,024, and the number of reviews posted was 331,084. For each brand b, we regarded the point k scored by a user v as the opinion k of v, and constructed the opinion diffusion sequence $\mathcal{D}_{T_0}(b)$ consisting of 3-tuple (k, t, v). In particular, we focused on these brands in which the number of samples $N = |\mathcal{D}_{T_0}(b)|$ was greater than 500. Then, the number of brands was 120. We refer to this dataset as the @cosme dataset.

5.2 Results

We applied the proposed method to the @cosme dataset. Again, we adopted the temporal decay voter models with the exponential and the power-law distributions, and used the significance level $\alpha = 0.005$ for model selection. There were 9 brands such that $2Y_N^e > \chi_{1,\alpha}$, and 93 brands such that $2Y_N^p > \chi_{1,\alpha}$. Here, in the same way as the previous section, Y_N^e and Y_N^p denote the log-likelihood ratio statistics of the exponential and the power-law TDV models against the base TDV model, respectively.

[4] http://www.cosme.net/

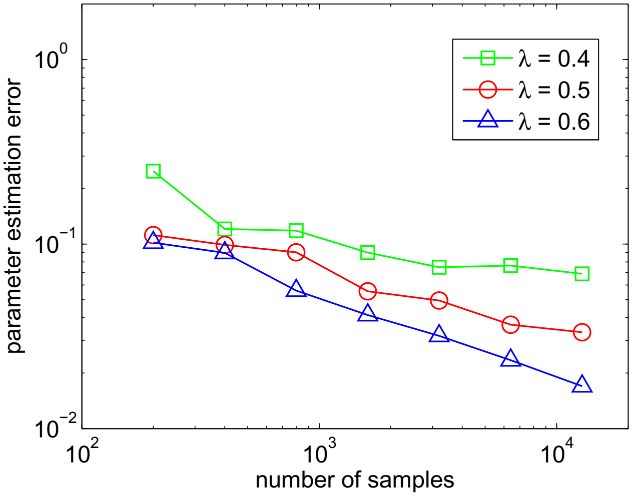


Fig. 4. Results of Parameter estimation error for the power-law TDV model

Further, there were 92 brands such that $2Y_N^p > \max\left(\chi_{1,\alpha}, 2Y_N^e\right)$, one brand such that $2Y_N^e > \max\left(\chi_{1,\alpha}, 2Y_N^p\right)$, and 27 brands such that $\max\left(2Y_N^p, 2Y_N^e\right) \leq \chi_{1,\alpha}$. Namely, according to the proposed method, 92 brands were the power-law TDV model, 27 brands were the base TDV model, and only one brand was the exponential TDV model. These results show that most brands conform to the power-law TDV model. This also agrees with the work [1,19] that many human actions are related to power-laws.

Figures 5 and 6 show the results for the @cosme dataset from the point of view of the power-law TDV model. Figure 5 plots the log-likelihood ratio statistic Y_N^p for each brand as a function of sample size N, where the thick solid line indicates the value of $\chi_{i,\alpha}$. In addition to the brands plotted, there is a brand such that $Y_N^p = Y_N^e = 0$. It was brand "YOJIYA", which is a traditional Kyoto brand, and is known as a brand releasing new products less frequently. Thus, we speculate that it conforms to the base TDV model. Figure 6 plots the pair $\left(Y_N^p, \hat{\lambda}(N)\right)$ for the brands in which the power-law TDV model was selected by the proposed method, where $\hat{\lambda}(N)$ is the value of parameter λ estimated by the proposed method from the observed data $\mathcal{D}_{T_0}(b)$ of $|\mathcal{D}_{T_0}(b)| = N$. From Figure 6, we observe that Y_N^p and $\hat{\lambda}(N)$ are positively correlated. This agrees with the fact that the power-law TDV model with $\lambda = 0$ corresponds to the base TDV model. In Figures 5 and 6, the big solid red circle indicates the brand "LUSH-JAPAN", which had the largest values of Y_N^p, $\hat{\lambda}(N)$ and N, respectively. We also find the big solid green triangle in Figure 5 as a brand that had a large value of Y_N^p and a relatively small value of N. This was the brand "SHISEIDO ELIXIR SUPERIEUR", which had the seventh largest value of Y_N^p, $N = 584$, and $\hat{\lambda}(N) = 0.58$. Note that these brands "LUSH-JAPAN" and "SHISEIDO ELIXIR SUPERIEUR" are known as brands that were recently established and release new products frequently. Thus, we speculate that they conform to the power-law TDV model with large λ.

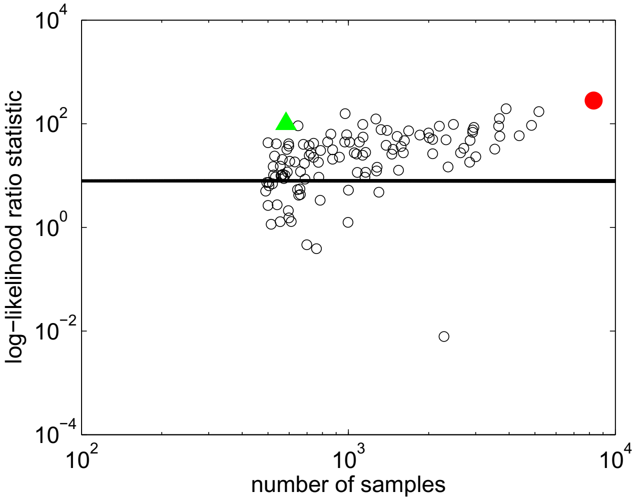


Fig. 5. Log-likelihood ratio statistic Y_N^p and number of samples N for the @cosme dataset

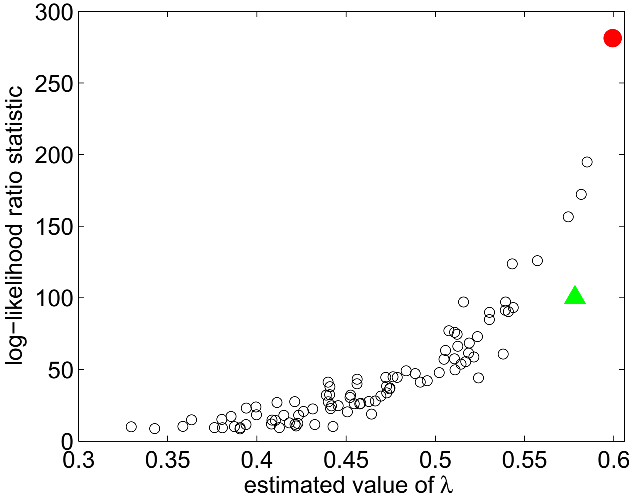


Fig. 6. Log-likelihood ratio statistic Y_N^p and estimated parameter value $\hat{\lambda}(N)$ for the @cosme dataset

6 Conclusion

We addressed the problem of how people make their own decisions based on their neighbors' opinions. The model best suited to discuss this problem is the voter model and several variants of this model have been proposed and used extensively. However, all of these models assume that people use their neighbors' latest opinions. People change opinions over time and some opinions are more persistent and some others are less persistent. These depend on many factors but the existing models do not take this effect into consideration. In this paper, we, in particular, addressed the problem of how people's opinions are affected by their own and other peoples' opinion histories. It would be reasonable to assume that older opinions are less influential and recent ones are more influential. Based on this assumption, we devised a new voter model, called the temporal decay voter (TDV) model which uses all the past opinions in decision making in which decay is assumed to be a linear combination of representative decay functions each with different decay factors. The representative functions include the linear decay, the exponential decay, the power-law decay and many more. Each of them specifies only the form and the parameters remain unspecified. We formulated this as a machine learning problem and solved the following two problems: 1) Given the observed sequence of people's opinion manifestation and an assumed decay function, learn the parameter values of the function such that the corresponding TDV model best explains the observation, and 2) Given a set of decay functions each with the optimal parameter values, choose the best model and refute others. We solved the former problem by maximizing the likelihood and derived an efficient parameter updating algorithm, and the latter problem by choosing the decay model that maximizes the log likelihood ratio statistic. We first tested the proposed algorithms by synthetic datasets assuming that there are two decay models: the exponential decay and the power-law decay. We confirmed that the learning algorithm correctly identifies the parameter values and the model selection algorithm correctly identifies which model the data came from. We then applied the method to the real opinion diffusion data taken from a Japanese word-of-mouth communication site for cosmetics. We used the two decay functions above and added no decay function as a baseline. The result of the analysis revealed that opinions of most of the brands conform to the TDV model of the power-law decay function. We found this interesting because this is consistent with the observation that many human actions are related to the power-law. Some brands showed behaviors characteristic to the brands, e.g., the older brand that releases new product less frequently naturally follows no decay TDV and the newer brand that releases new product more frequently naturally follows the power-law decay TDV with large decay constant, which are all well interpretable.

Acknowledgments. This work was partly supported by Asian Office of Aerospace Research and Development, Air Force Office of Scientific Research under Grant No. AOARD-11-4111, and JSPS Grant-in-Aid for Scientific Research (C) (No. 23500194).

References

1. Barabási, A.L.: The origin of bursts and heavy tails in human dynamics. Nature 435, 207–211 (2005)
2. Castellano, C., Munoz, M.A., Pastor-Satorras, R.: Nonlinear *q*-voter model. Physical Review E 80, Article 041129 (2009)
3. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009), pp. 199–208 (2009)

4. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: Proceedings of the 10th IEEE International Conference on Data Mining (ICDM 2010), pp. 88–97 (2010)
5. Crandall, D., Cosley, D., Huttenlocner, D., Kleinberg, J., Suri, S.: Feedback effects between similarity and social influence in online communities. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), pp. 160–168 (2008)
6. Domingos, P.: Mining social networks for viral marketing. IEEE Intelligent Systems 20, 80–82 (2005)
7. Even-Dar, E., Shapira, A.: A Note on Maximizing the Spread of Influence in Social Networks. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 281–286. Springer, Heidelberg (2007)
8. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. SIGKDD Explorations 6, 43–52 (2004)
9. Holme, P., Newman, M.E.J.: Nonequilibrium phase transition in the coevolution of networks and opinions. Physical Review E 74, Article 056108 (2006)
10. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003), pp. 137–146 (2003)
11. Kimura, M., Saito, K., Nakano, R., Motoda, H.: Extracting influential nodes on a social network for information diffusion. Data Mining and Knowledge Discovery 20, 70–97 (2010)
12. Kimura, M., Saito, K., Ohara, K., Motoda, H.: Learning to predict opinion share in social networks. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), pp. 1364–1370 (2010)
13. Kimura, M., Saito, K., Ohara, K., Motoda, H.: Detecting Anti-majority Opinionists Using Value-Weighted Mixture Voter Model. In: Elomaa, T., Hollmén, J., Mannila, H. (eds.) DS 2011. LNCS, vol. 6926, pp. 150–164. Springer, Heidelberg (2011)
14. Koren, Y.: Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009), pp. 447–456 (2009)
15. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. ACM Transactions on the Web 1, Article 5 (2007)
16. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007), pp. 420–429 (2007)
17. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
18. Newman, M.E.J., Forrest, S., Balthrop, J.: Email networks and the spread of computer viruses. Physical Review E 66, Article 035101 (2002)
19. Oliveira, J.G., Barabási, A.L.: Dawin and Einstein correspondence patterns. Nature 437, 1251 (2005)
20. Sood, V., Redner, S.: Voter model on heterogeneous graphs. Physical Review Letters 94, Article 178701 (2005)
21. Wu, F., Huberman, B.A.: How Public Opinion Forms. In: Papadimitriou, C., Zhang, S. (eds.) WINE 2008. LNCS, vol. 5385, pp. 334–341. Springer, Heidelberg (2008)
22. Yang, H., Wu, Z., Zhou, C., Zhou, T., Wang, B.: Effects of social diversity on the emergence of global consensus in opinion dynamics. Physical Review E 80, Article 046108 (2009)

Viral Marketing for Product Cross-Sell through Social Networks

Ramasuri Narayanam and Amit A. Nanavati

IBM Research, India
{nramasuri,namit}@in.ibm.com

Abstract. The well known *influence maximization problem* [1] (or viral marketing through social networks) deals with selecting a few influential initial seeds to maximize the awareness of product(s) over the social network. In this paper, we introduce a novel and generalized version of the influence maximization problem that considers simultaneously the following three practical aspects: (i) Often cross-sell among products is possible, (ii) Product specific costs (and benefits) for promoting the products have to be considered, and (iii) Since a company often has budget constraints, the initial seeds have to be chosen within a given budget. We refer to this generalized problem setting as *Budgeted Influence Maximization with Cross-sell of Products (B-IMCP)*. To the best of our knowledge, we are not aware of any work in the literature that addresses the B-IMCP problem which is the subject matter of this paper. Given a fixed budget, one of the key issues associated with the B-IMCP problem is to choose the initial seeds within this budget not only for the individual products, but also for promoting cross-sell phenomenon among these products. In particular, the following are the specific contributions of this paper: (i) We propose an influence propagation model to capture both the cross-sell phenomenon and product specific costs and benefits; (ii) As the B-IMCP problem is NP-hard computationally, we present a simple greedy approximation algorithm and then derive the approximation guarantee of this greedy algorithm by drawing upon the results from the theory of matroids; (iii) We then outline two efficient heuristics based on well known concepts in the literature. Finally, we experimentally evaluate the proposed approach for the B-IMCP problem using a few well known social network data sets such as WikiVote data set, Epinions, and Telecom call detail records data.

Keywords: Social networks, influence maximization, cross-sell, budget constraint, matroid theory, costs, benefits, and submodularity.

1 Introduction

The phenomenon of viral marketing is to exploit the social connections among the individuals to promote awareness for new products [2–4]. One of the key issues in viral marketing through social networks is to select a set of influential seed members (also called as *initial seeds*) in the social network and give them free samples of the product (or provide promotional offers) to trigger cascade of influence over the network [5]. The problem is, given an integer k, how should we choose a set of k initial seeds so that the cascade of influence over the network is maximized? This problem is known as

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 581–596, 2012.

influence maximization problem [1]. This problem is shown to be a NP-hard problem in the context of certain information diffusion models such as linear threshold model and independent cascade model [1]. Influence maximization problem is well studied in the literature [5, 6, 1, 7–15] in the context of a single product and multiple independent products; and we refer to the section on the relevant work for more details.

However, the existing work in the literature on viral marketing of products through social networks ignores the following important aspects that we often experience in several practical settings:

- *Cross-sell Phenomenon:* Certain products are complementary to each other in the sense that there is a possibility of cross-sell among these products,
- *Product Specific Costs and Benefits:* There is a cost associated with each product in order to provide promotional offers (or discounts) to each of the initial seeds. Similarly, there is a benefit associated with each product, when someone buys that product, and
- *Budget Constraint:* Companies have often have budget constraints and hence the initial seeds have to be chosen within a given budget.

To the best of our knowledge, we are not aware of any work in the literature that simultaneously deals with the cross-sell phenomenon, the product specific costs and benefits, and the budget constraint while addressing the influence maximization problem. We address this generalized version of the influence maximization problem in this paper.

In particular, we consider the following framework. Let P_1 be a set of t_1 independent products and similarly P_2 be another set of t_2 independent products. We assume that cross-sell is possible from the products in P_1 to the products in P_2 and, in particular, we consider the following specific form for the cross-sell phenomenon. The need for buying any product in P_2 arises only after buying some product in P_1. A real life instance of this type of cross-sell is as follows. Consider the context of computers and printers; in general, the need for buying any printer arises after buying a computer. Informally, cross-sell is the action or practice of selling an additional product (or service) to an existing customer. Typically, the additional product (i.e. in P_2) is of interest to the customer only because of a prior product (i.e. in P_1) purchased by the customer. From a social network diffusion model standpoint, the purchase of products in P_1 causes a lowering of threshold for buying certain products in P_2. It is this phenomenon that we explore in this paper.

We consider different costs and benefits for each product in P_1 and P_2. Since companies owning the products often have budget constraints, we can offer free samples (or promotional offers) of the products to the initial seeds within a given budget B. In this setting, one of the key issues is to choose the initial seeds not only for each individual product in P_1, but also for promoting the cross-sell phenomenon from the products in P_1 to the products in P_2. We refer to the above problem of selecting, within budget B, a set of initial seeds to maximize influence through the social network and hence to maximize the revenue to the company as the *budgeted influence maximization with cross-sell of products (B-IMCP)* problem. In what follows, we highlight the main challenges associated with the B-IMCP problem that make it non-trivial to address and also summarize the main contributions of this paper:

(A) Modeling Aspects: How to model the propagation of influence in social networks in the context of cross-sell of products? We note that the variants of the linear threshold model [16, 17, 1, 14] in their current form are not sufficient to model the cross-sell phenomenon.

In this paper, we address this issue by proposing a simple model of influence propagation over social networks in the context of cross-sell of products by naturally extending the well known linear threshold model [1]. We call this *linear threshold model for cross-sell of products (LT-CP)*. We then prove a few important properties of the LT-CP model such as monotonicity and submodularity.

(B) Algorithmic Aspects: We note that the B-IMCP problem, in the context of the LT-CP model, turns out to be a computationally hard problem, i.e. NP-hard (see Section 3 for more details). This calls for designing an approximation algorithm to address the B-IMCP problem. In this paper, we propose a greedy approximation algorithm to address the B-IMCP problem. Assume that B is the budget for the company. Let c_M^1 and c_M^2 be the maximum cost of providing a free sample of any product in P_1 and P_2 respectively. On similar lines, let c_m^1 and c_m^2 be the minimum cost of providing a free sample of any product in P_1 and P_2 respectively. We show that the approximation guarantee of the proposed greedy algorithm for the B-IMCP problem is $\frac{Bc_m}{B(c_M+c_m)+c_M c_m}$, where $c_M = max(c_M^1, c_M^2)$ and $c_m = min(c_m^1, c_m^2)$. Interestingly, the approximation factor of the greedy algorithm is independent of the number of products and it only depends on (a) the budget B, and (b) the maximum and the minimum costs to provide free samples of products in P_1 and P_2 respectively.

We use the techniques from the theory of submodular function maximization over Matroids [18] to derive the approximation guarantee of the proposed greedy algorithm. We must note that the body of relevant work in the literature works with the framework of *approximations for maximizing submodular set functions [19]* to derive the approximation guarantee of the algorithms for the variants of the influence maximization problem with a single product and multiple independent products [1, 7, 14]. However, these techniques are not sufficient for the B-IMCP problem setting as (i) we have to work with cross-sell of products, (ii) product specific costs and benefits, and (ii) we have to deal with the budget constraint.

(C) Experimental Aspects: We experimentally observe that the proposed greedy approximation algorithm for the B-IMCP problem runs slow even on moderate size network data sets. We must note that similar observations are reported in the literature in the context of the greedy algorithm [1] for the well known influence maximization problem [7, 8]. The existing scalable and efficient heuristics [7, 8] for the influence maximization problem do not directly apply to the context of the B-IMCP problem. How to alleviate this scalability issue of determining a solution for the B-IMCP problem on large social network data sets? In this paper, we present an efficient heuristic for the B-IMCP problem. We experimentally evaluate the performance of the greedy approximation and heuristic algorithms using experimentation on several social network data sets such as WikiVote trust network, Epinions trust network, and Telecom call detail records data. We also compare and contrast the performance of the greedy approximation and the heuristic algorithms with that of two well known benchmark heuristics.

1.1 Novelty of This Paper

The primary contribution and the novelty of this paper is three fold: (i) Introducing the phenomenon of cross-sell of products and product specific costs and benefits while addressing the influence maximization problem, (ii) Naturally extending the well known linear threshold model to capture the cross-sell phenomenon, and (iii) Performing non-trivial analysis of the simple greedy algorithm for the B-IMCP problem to derive the approximation guarantee of the same.

2 Relevant Work

There are two well known operational models in the literature that capture the underlying dynamics of the information propagation in the viral marketing process. They are the linear threshold model [17, 16, 1] and the independent cascade model [20, 1]. In the interest of space constraints, we only present the most relevant work on the influence maximization problem in the literature and we categorize this into three groups as follows.

Influence Maximization with Single Product: Domingos and Richardson [5] and Richardson and Domingos [6] were the first to study influence maximization problem as an algorithmic problem. Computational aspects of the influence maximization problem are investigated by Kempe, Kleinberg, and Tardos [1] and they showed that the optimization problem of selecting the most influential nodes is NP-hard. The authors proposed a greedy approximation algorithm for the influence maximization problem.

Leskovec, et. al. [7] proposed an efficient algorithm for the influence maximization problem based on the submodularity of the underlying objective function that scales to large problems and is reportedly 700 times faster than the greedy algorithm of [1] and later Chen, Wang, and Yang [8] further improved this result. Even-Dar and Shapira [11], Kimura and Saito [9], Mathioudakis *et.al.* [10], Ramasuri and Narahari [21] considered various interesting extensions to the basic version of the influence maximization problem in social networks.

Influence Maximization with Multiple Products: Recently, Datta, Majumder, and Shrivastava [12] considered the influence maximization problem for *multiple independent products*.

Viral Marketing with Competing Companies: Another important branch of the research work in viral marketing is to study the algorithmic problem of how to introduce a new product into the market in the presence of a single or multiple competing products already in the market [13–15].

3 The Proposed Model for the B-IMCP Problem

Here we first present the LT-CP model for the B-IMCP problem. We must note that this model is a natural extension of the well known linear threshold model [17, 16, 1].

3.1 The LT-CP Model

Let $G = (V, E)$ be a directed graph representing an underlying social network where V represents a set of individuals and E is a set of directed edges representing friendship relations among these individuals. Let $|V| = n$ and $|E| = m$. For each edge $(i, j) \in E$, we define a weight w_{ij} and this indicates the probability with which the available information at individual i passes to individual j. Another interpretation of this weight w_{ij} is the probability with which individual j is influenced by the recommendation of the individual i. If there is no confusion, here onwards, we refer to individuals and nodes interchangeably. Similarly, we also refer to graphs and networks interchangeably.

In this context, we consider the following setting as introduced in Section 1. Note that $P_1 = \{1, 2, \ldots, t_1\}$ is a set of t_1 independent products and similarly $P_2 = \{1, 2, \ldots, t_2\}$ is another set of t_2 independent products. Cross-sell is possible from the products in P_1 to the products in P_2. For the simplicity of the technical analysis, we assume that (i) $t_1 = t_2$ and call this common value t; and (ii) there exists an onto function from P_1 to P_2, call it $H : P_1 \to P_2$, such that for each product $k \in P_1$ there exists exactly one product in P_2 (namely $H(k)$). That is, for each product $k \in P_1$, there exists a product $H(k) \in P_2$ such that cross-sell is possible from product $k \in P_1$ to product $H(k) \in P_2$. A company, call it M, owns the products in P_1 and P_2 and it has a fixed budget B for conducting viral marketing campaign for these products.

In particular, a free sample (or promotional offer) of product k in P_1 incurs a cost of c_k^1 and similarly a free sample (or promotional offer) of product z in P_2 incurs a cost of c_z^2. Also, when an item of product k in P_1 is sold, it leads to a benefit b_k^1 to the company. On the similar lines, when an item of product z in P_2 is sold, it leads to a benefit b_z^2 to the company. Now, we define a few important notations and terminology as follows:

- We call an individual (and is represented by a *node* in the graph) in the network *active* if he/she buys any product in P_1 or P_2, and *inactive* otherwise. For each node $i \in V$, let N_i be the set of its neighbors. Node i is influenced by any neighbor j according to a weight w_{ji}. Assume these weights are normalized in such a way that $\sum_{j \in N_i} w_{ji} \le 1$.
- We define the following for each node $i \in V$. For each product $k \in P_1$ and for each $i \in V$, we define A_i^k to be the set of node i's active neighbors who bought product $k \in P_1$. On similar lines, for each product $z \in P_2$ and for each $i \in V$, we define A_i^z to be the set of node i's active neighbors who have initially bought product $H(z) \in P_1$ and then bought product $z \in P_2$.

We now define when the individual nodes buy the products in P_1 and P_2. Recall that the products in P_1 are independent. The decision of a node $i \in V$ to buy product $k \in P_1$ is based on a threshold function (f_i^k), which is dependent on N_i and a threshold (θ_i^k) chosen uniformly at random by node i from the interval $[0, 1]$. This threshold represents the minimum amount of influence required from the active neighbors of node i (who bought product $k \in P_1$) in order for node i to become active. Note that $f_i^k : 2^{N_i} \to [0, 1]$ is defined as $f_i^k(S) = \sum_{j \in S} w_{ji}, \forall S \subseteq N_i$. Now, we say that node i buys product $k \in P_1$ if $f_i^k(A_i^k) \ge \theta_i^k$.

Recall that cross-sell is possible from the products in P_1 to the products in P_2 and this cross-sell relationship is defined using the function H which is onto. For each $z \in P_2$

and for each $i \in V$, we initially set the threshold θ_i^z of node i for buying the product z to be a very high quantity to model the fact that no node $i \in V$ buys product z in P_2 before buying product $H(z)$ in P_1. Now assume that node $i \in V$ has bought the product $H(z) \in P_1$ and since cross-sell is possible from product $H(z)$ to product $z \in P_2$, we decrease the threshold θ_i^z by defining that θ_i^z is chosen uniformly at random from the interval $[0, a]$ where $0 \leq a < 1$. Now the decision of node i to buy product $z \in P_2$ is based on a threshold function (f_i^z), which is dependent on N_i and the threshold θ_i^z. This threshold represents the minimum amount of influence required from the active neighbors of node i (who have initially bought product $H(z) \in P_1$ and then bought product $z \in P_2$) in order for node i to become active. Note that $f_i^z : 2^{N_i} \rightarrow [0, 1]$ is defined as $f_i^z(S) = \sum_{j \in S} w_{ji}, \forall S \subseteq N_i$. Now, we say that node i buys product $z \in P_2$ if $f_i^z(A_i^z) \geq \theta_i^z$.

3.2 The B-IMCP Problem

In the presence of the above model, we now define the following. For each product $k \in P_1$, let S_1^k be the initial seed set. We define the influence spread of the seed set S_1^k, call it $\Gamma(S_1^k)$, to be the expected number nodes that buy the product $k \in P_1$ at the end of the diffusion process, given that S_1^k is the initial seed set. On similar lines, for each product $z \in P_2$, let S_2^z be the initial seed set. We define the influence spread of the seed set S_2^z, call it $\Delta(S_2^z)$, to be the expected number of nodes that initially buy the product $H(z) \in P_1$ and then buy the product $z \in P_2$ at the end of the diffusion process, given that S_2^z is the initial seed set. For any specific choice of the initial seed sets S_1^k for each $k \in P_1$ and S_2^z for each $z \in P_2$, we now define the objective function, call it $\sigma(S_1^1, \ldots, S_1^t, S_2^1, \ldots, S_2^t)$, to be the expected revenue to the company at the end of the diffusion process. That is,

$$\sigma(S_1^1, \ldots, S_1^t, S_2^1, \ldots, S_2^t) = \sum_{k \in P_1} \Gamma(S_1^k) b_k^1 + \sum_{z \in P_2} \Delta(S_2^z) b_z^2. \tag{1}$$

Given the budget B, the B-IMCP problem seeks to find the initial seed sets $S_1^1, \ldots, S_1^t$, $S_2^1, \ldots, S_2^t$ such that the objective function is maximized.

We now show that the B-IMCP problem is a computationally hard problem.

Lemma 1. *The B-IMCP problem in the presence of the LT-CP model is NP-hard.*

Proof. By setting $|P_1| = t = 1$, $P_2 = \phi$, $c_k^1 = 1$ for each $k \in P_1$, and $b_k^1 = 1$ for each $k \in P_1$, we get that any arbitrary instance of the B-IMCP problem with the LT-CP model reduces exactly to an instance of the influence maximization problem with the linear threshold model [1]. It is already known that the influence maximization problem with the linear threshold model is NP-hard [1]. ■

3.3 Properties of the Proposed Model

We show that the objective function $\sigma(.)$ is monotone and submodular. The proof of monotonicity of $\sigma(.)$ is immediate, as the expected number of individuals that buy the product(s) does not decrease when we increase the number of initial seeds. Thus we now focus on the proof of submodularity of $\sigma(.)$ and we note that this can be proved easily using the results from Kempe *et al.* [1]. However, for the sake of completeness, we give a sketch of the proof of this result.

Lemma 2. *For any arbitrary instance of the LT-CP model, the objective function* $\sigma(.)$ *is submodular.*

Proof Sketch: There are two main steps in this proof. First, for each $k \in P_1$, we have to show that $\Gamma(S_1^k)$ is submodular. Second, for each $z \in P_2$, we have to show that $\Delta(S_2^z)$ is submodular. Recall that (i) multiplying a submodular function with a positive constant results in a submodular function; and (ii) the sum of submodular functions is also a submodular function. This implies that the objective function $\sigma(.)$ is submodular.

Since the products in P_1 are independent, for each $k \in P_1$, it is straight forward to see that $\Gamma(S_1^k)$ is submodular due to Theorem 2.5 in Kempe *et al.* [1].

Claim 2**:** $\Delta(S_2^z)$ is submodular.
Since cross-sell is possible from the product $H(z) \in P_1$ to the product $z \in P_2$, we can compute $\Delta(S_2^z)$ after the diffusion of $H(z)$ finishes. We model the spread of $H(z)$ using the technique of live-edge paths as in Theorem 2.5 in [1]. Suppose that every node i picks at most one of its incoming edges at random, selecting the edge from neighbor j with probability w_{ji} and selecting no edge with probability $1 - \sum_j w_{ji}$. Each such selected edge in G is declared to be *live* and all other edges are declared to be *blocked*. Let $G^{'}$ be the graph obtained from the original graph G by retaining only the live-edges and let $\Pi(G)$ be the set of all such $G^{'}$. Let $P(G^{'}$ be the probability of obtaining $G^{'}$ from G using the process of live edges. Note that each $G^{'}$ models a possible trace of the spread of the product $H(z) \in P_1$.

Now consider $G^{'} \in \Pi(G)$ and a node i in $G^{'}$. Recall that each node i in the original graph G picks at most one of its incoming edges at random, selecting the edge from neighbor j with probability w_{ji} and selecting no edge with probability $1 - \sum_j w_{ji}$. For this reason, each node i in $G^{'}$ has at most one incoming edge. Now if node i in $G^{'}$ has an incoming edge, call it from node j, then i picks this only incoming edge with probability w_{ji} and picks no edge with probability $1 - w_{ji}$. Each such selected edge in $G^{'}$ is declared to be *live* and all the other edges are declared to be *blocked*. Using the arguments exactly similar to that in Kempe *et al.* [1], it turns out that proving this claim is equivalent to reachability via live-edge paths in $G^{'}$. Let $G^{''}$ be the graph obtained from $G^{'}$ by retaining only the live-edges and assume that $\Pi(G^{'})$ is the set of all such $G^{''}$. Let $P(G^{''})$ be the probability of obtaining $G^{''}$ from $G^{'}$ using the process of live edges. For each $G^{''} \in \Pi(G^{'})$, we define $a_{G^{''}}(S_2^z, m)$ to be the number of nodes that are exactly m steps away on any path starting from some node in S_2^z in $G^{''}$; i.e. $a_{G^{''}}(S_2^z, m) = |\{v \in V \mid d_{G^{''}}(S_2^z, v) = m\}|$ where $d_{G^{''}}(S_2^z, v)$ is the length of the shortest distance from any node in S_2^z to v. We can now write $\Delta(S_2^z)$ as follows:

$$\Delta(S_2^z) = E_{G^{'} \in \Pi(G)}\Big[E_{G^{''} \in \Pi(G^{'})}\Big[\sum_{m=0}^{\infty} a_{G^{''}}(S_2^z, m)\Big]\Big] \tag{2}$$

$$= \sum_{G^{'} \in \Pi(G)} \sum_{G^{''} \in \Pi(G^{'})} P(G^{'})P(G^{''}) \sum_{m=0}^{\infty} a_{G^{''}}(S_2^z, m) \tag{3}$$

Now let us define $h(S_2^z, G^{''}) = \sum_{m=0}^{\infty} a_{G^{''}}(S_2^z, m)$ for all $S_2^z \subseteq V$. If we can show that $h(.)$ is submodular, then $\Delta(S_2^z)$ is also submodular as it is a non-negative linear combination of $h(.,.)$. It is not difficult to show that $h(.)$ is submodular and, in the interest of space, we do not present the proof of the same. ■

Note: It is important to note that the submodularity of $\sigma(.)$ may break down if we change the way to model the cross-sell relationships among the products.

4 Greedy Approximation Algorithm for the B-IMCP Problem

Motivated by the greedy algorithm [1] for the well known influence maximization problem, we now present a simple greedy algorithm for the B-IMCP problem. Let $S_1^1, S_1^2, \ldots, S_1^t$ be the sets of initial seeds for the t products in P_1 respectively. Let $S_2^1, S_2^2, \ldots, S_2^t$ be the sets of initial seeds for the t products in P_2 respectively. Initially set $S_1^k = \phi$ for each $k \in P_1$ and $S_2^z = \phi$ for each $z \in P_2$. Algorithm 1 presents the proposed greedy algorithm and the following is the main idea of the same. The algorithm runs in iterations until the budget B is exhausted to select the initial seeds. The following is performed in each iteration of the algorithm:

Algorithm 1. Greedy Algorithm to Select the Initial Seeds for the B-IMCP Problem

```
Initially set $S_1^k = \phi \quad \forall k \in P_1$ and $S_2^z = \phi \quad \forall z \in P_2$.
while $B > 0$ do
    Pick a node $v_1 \in V \setminus \cup_{k \in P_1} S_1^k$ such that $v_1$ maximizes $valx = \frac{\sigma\left(\cup_{k \in P_1} S_1^k \bigcup \{v_1\} \bigcup \cup_{z \in P_2} S_2^z\right) - \sigma\left(\cup_{k \in P_1} S_1^k \bigcup \cup_{z \in P_2} S_2^z\right)}{c_i^1}$, when we active it with some product $i \in P_1$
    Pick a node $v_2 \in V \setminus \cup_{z \in P_2} S_2^z$ such that $v_2$ maximizes $valy = \frac{\sigma\left(\cup_{k \in P_1} S_1^k \bigcup \cup_{z \in P_2} S_2^z \bigcup \{v_2\}\right) - \sigma\left(\cup_{k \in P_1} S_1^k \bigcup \cup_{z \in P_2} S_2^z\right)}{c_j^2}$, when we active it with some product $j \in P_2$.
    if $valx \geq valy$ and $B - c_i^1 \geq 0$ then
        Set $S_1^i \leftarrow S_1^i \cup \{v_1\}$, and $B \leftarrow B - c_i^1$
        As cross-sell is possible from from product $i \in P_1$ to product $H(i) \in P_2$, update the value of $\theta_{v_1}^{H(i)}$ for node $v_1$
    end if
    if $valy > valx$ and $B - c_j^2 \geq 0$ then
        Set $S_2^j \leftarrow S_2^j \cup \{v_2\}$, and $B \leftarrow B - c_j^2$
    end if
end while
Return $\left(S_1^1, S_1^2, \ldots, S_1^t, S_2^1, S_2^2, \ldots, S_2^t\right)$ as the initial seed set
```

(i) Let $v_1 \in V \setminus \cup_{k \in P_1} S_1^k$ be the next best seed for P_1 in the sense that when we activate it with product $i \in P_1$, v_1 maximizes the ratio of increase in the expected revenue gain to the cost c_i^1; that is, v_1 maximizes $\frac{\sigma\left(\cup_{k \in P_1} S_1^k \bigcup \{v_1\} \bigcup \cup_{z \in P_2} S_2^z\right) - \sigma\left(\cup_{k \in P_1} S_1^k \bigcup \cup_{z \in P_2} S_2^z\right)}{c_i^1}$

and call this $valx$;
(ii) Let $v_2 \in V \backslash \cup_{z \in P_2} S_2^z$ be the next best seed in the sense that when we activate it with some product $j \in P_2$, v_2 maximizes the ratio of increase in the expected revenue gain to the cost c_j^1; that is, v_2 maximizes $\frac{\sigma\left(\cup_{k \in P_1} S_1^k \bigcup \cup_{z \in P_2} S_2^z \bigcup \{v_2\}\right) - \sigma\left(\cup_{k \in P_1} S_1^k \bigcup \cup_{z \in P_2} S_2^z\right)}{c_j^2}$ and call this $valy$;
(iii) If $valx \geq valy$ and $B - c_i^1 \geq 0$, then we add v_1 to the set of seeds for S_1^i and also decrease B by an amount c_i^1. To take care of the cross-sell phenomenon, we also update the threshold $\theta_{v_1}^{H(i)}$ for node v_1;
(iv) If $valy > valx$ and $B - c_j^2 \geq 0$, then we add v_2 to the set S_2^j and decrease B by an amount c_j^2.

Finally, the greedy algorithm returns $\left(S_1^1, S_1^2, \ldots, S_1^t, S_2^1, S_2^2, \ldots, S_2^t\right)$ as the initial seed set for the B-IMCP problem.

Running Time of Algorithm 1. Let $c = min\{c_1^1, c_2^1, \ldots, c_t^1, c_1^2, c_2^2, \ldots, c_t^2\}$ and $t = \frac{B}{c}$. Note that Algorithm 1 runs at most t rounds. In each iteration of this algorithm, we have to check at most $O(n)$ nodes to determine the best seed for P_1 and P_2. To determine $valx$ (or $valy$) in each iteration, we have to essentially count the number of nodes that are reachable from the initial seed elements using the live edges in the graph and it takes at most $O(m)$ time where m is the number of edges. Also, as underlying information diffusion process is a stochastic process, we have to repeat the experiment a number of times (call it R times) and take the average to determine values for $valx$ and $valy$ in each iteration. With all this in place, the running time of Algorithm 1 is $O(tnRm)$ where $t = \frac{B}{c}$.

4.1 Analysis of Algorithm 1

We now analyze Algorithm 1 and derive the approximation guarantee of the same. Our analysis uses results from matroid theory and Calinescu, *et al.* [18]. Towards this end, we first recall the definition of a matroid.

Matroid: A Matroid is a pair $M = (U, I)$, where $I \subseteq 2^U$ is a subset of the power set (all possible subsets) of U that satisfies the following constraints:

- I is an *independent set system*: $\phi \in I$ and $A \in I$, any set $B \subset A$ then $B \in I$ (All subsets of any independent set is also independent).
- $\forall A, B \in I$ and $|A| > |B|$: $\exists x \in A - B$ s.t. $B \cup \{x\} \in I$.

The first constraint defines an independent set system, and each set $S \in I$ is called an independent set. The problem of maximizing a sub-modular function on a Matroid is to find the independent set $S \in I$, s.t. $f(S)$ is maximum over all such sets S. If the input set is a Matroid, it is known that the sub-modular function maximization can be approximated within a constant factor ($\frac{1}{2}$ or $(1-1/e)$ in certain cases) using the greedy hill climbing approach (Nemhauser, Wolsey, and Fisher [19]).

The independent set system essentially defines the feasible sets over which the objective function is defined. In the context of B-IMCP problem, we call an initial seed set $\left(S_1^1, S_1^2, \ldots, S_1^t, S_2^1, S_2^2, \ldots, S_2^t\right)$ *feasible*, when the sum of costs of providing free samples of the products in this initial seed set is within the budget B. It is easy to see that the feasible seed sets form an Independent set system, I. However, the feasible seed sets in I do not form a matroid since they can violate the second condition in the definition of matroid due to the budget constraint. The following example validates this fact.

Example 1. Consider a graph with 10 individuals, i.e. $V = \{1,2,3,4,5,6,7,8,9,10\}$. There are 4 products in P_1; i.e., $P_1 = \{t_1^1, t_2^1, t_3^1, t_4^1\}$. There are 4 products in P_2; i.e., $P_2 = \{t_1^2, t_2^2, t_3^2, t_4^2\}$. Also let $B = 20$, $c_1^1 = 4$, $c_2^1 = 3$, $c_3^1 = 5$, $c_4^1 = 6$, $c_1^2 = 2$, $c_2^2 = 4$, $c_3^2 = 2$, and $c_4^2 = 4$. Now consider two feasible initial seed sets as follows:

Consider a feasible seed set $\left(S_1^1, S_1^2, \ldots, S_1^4, S_2^1, S_2^2, \ldots, S_2^4\right)$ as follows. $S_1^1 = \phi$, $S_1^2 = \{2,3,4\}$, $S_1^3 = \{6\}$, $S_1^4 = \phi$, $S_2^1 = \phi$, $S_2^2 = \{2\}$, $S_2^3 = \{6\}$, $S_2^4 = \phi$. Note that the cost of providing the free samples of products in the initial seed set $\left(S_1^1, S_1^2, \ldots, S_1^4, S_2^1, S_2^2, \ldots, S_2^4\right)$ is $3+3+3+5+4+2 = 20$. Given that $B = 20$, it is clear that $\left(S_1^1, S_1^2, \ldots, S_1^4, S_2^1, S_2^2, \ldots, S_2^4\right)$ is a feasible seed set.

Consider another feasible seed set $\left(\bar{S}_1^1, \bar{S}_1^2, \ldots, \bar{S}_1^4, \bar{S}_2^1, \bar{S}_2^2, \ldots, \bar{S}_2^4\right)$ as follows. $\bar{S}_1^1 = \phi$, $\bar{S}_1^2 = \phi$, $\bar{S}_1^3 = \{3,9\}$, $\bar{S}_1^4 = \{6\}$, $\bar{S}_2^1 = \phi$, $\bar{S}_2^2 = \phi$, $\bar{S}_2^3 = \phi$, $\bar{S}_2^4 = \{6\}$. Note that the cost of providing the free samples of products in the initial seed set $\left(\bar{S}_1^1, \bar{S}_1^2, \ldots, \bar{S}_1^4, \bar{S}_2^1, \bar{S}_2^2, \ldots, \bar{S}_2^4\right)$ is $5+5+6+4 = 20$. Given that $B = 20$, it is clear that $\left(\bar{S}_1^1, \bar{S}_1^2, \ldots, \bar{S}_1^4, \bar{S}_2^1, \bar{S}_2^2, \ldots, \bar{S}_2^4\right)$ is a feasible seed set.

Note that the cardinality of the first feasible set is $|\{2,3,4,6\}| = 4$ and that of the second feasible set is $|\{3,6,9\}| = 3$. Moreover, observe that node 2 is an initial seed for product 2 in P_1 in the first feasible seed set and it is not an initial seed for any product in the second feasible seed set. However, we cannot add node 2 to the seed set of any product in P_1 and P_2 in the second feasible seed set without violating the budget constraint.

This example immediately leads to the following simple result.

Proposition 1. *For an arbitrary instance of the B-IMCP problem, the independent set system consisting of the feasible seed sets is not necessarily a Matroid.*

Hence we opt for a slightly weaker definition on an independent set system, called a p-system, defined as follows [12]. Informally, p-system says that for any set $A \subseteq V$, the sizes of any two maximal independent subsets of A do not differ by a factor more than p. Then according to Calinescu, *et al.* [18] the hill climbing gives a $\frac{1}{(p+1)}$ approximation of submodular function maximization for any p-system.

Towards this end, we first prove an useful result. Before proceeding further, we introduce the following notation. Assume that (i) c_M^1 is the maximum cost among all c_k^1, $k \in P_1$; i.e., $c_M^1 = max_{k \in P_1}\ c_k^1$, (ii) c_M^2 is the maximum cost among all c_z^2, $z \in P_2$; i.e., $c_M^2 = max_{z \in P_2}\ c_z^2$, (iii) c_m^1 is the minimum cost among all c_k^1, $k \in P_1$; i.e., $c_m^1 = min_{k \in P_1}\ c_k^1$, and (iv) c_m^2 is the minimum cost among all c_z^2, $z \in P_2$; i.e., $c_m^2 = min_{z \in P_2}\ c_z^2$.

Lemma 3. *The feasible seed sets for the B-IMCP problem form* $c_M\left(\frac{1}{c_m}+\frac{1}{B}\right)$*-system where* $c_M = max(c_M^1, c_M^2)$ *and* $c_m = min(c_m^1, c_m^2)$*.*

Proof. Consider an arbitrary instance of the B-IMCP problem. Note that V is the set of nodes in the graph G and let A be any subset of V. Let $\left(S_1^1, S_1^2, \ldots, S_1^4, S_2^1, S_2^2, \ldots, S_2^4\right)$ and $\left(\bar{S}_1^1, \bar{S}_1^2, \ldots, \bar{S}_1^4, \bar{S}_2^1, \bar{S}_2^2, \ldots, \bar{S}_2^4\right)$ be any two maximal feasible sets in A with maximum and minimum sizes respectively. Also let $S = S_1^1 \bigcup \ldots \bigcup S_1^t \bigcup S_2^1 \bigcup \ldots \bigcup S_2^t$ and $\bar{S} = \bar{S}_1^1 \bigcup \ldots \bigcup \bar{S}_1^t \bigcup \bar{S}_2^1 \bigcup \ldots \bigcup \bar{S}_2^t$. If $|S| \leq |\bar{S}|$, then S and $\bar{S}$ are of same size and hence the independent set system with all feasible seed sets forms a matroid. It is contradiction to Proposition 1 (see the Example 1). Hence we consider the case where $|\bar{S}| < |S|$. We will now bound how much the cardinality of S is greater than that of $\bar{S}$ in the worst case. We consider the following four cases.

Case 1 *(*$c_M^1 > c_M^2$ *and* $c_m^1 > c_m^2$ *):* The cardinality of S is much larger, in the worst case, than that of $\bar{S}$ when:

- All the seed elements in $\bar{S}$ are part of $\bar{S}_1^j$ for some product $j \in P_1$ such that $c_j^1 = c_M^1$.
- All initial seed elements of S are the initial seeds for some product $k \in P_2$ having cost $c_k^2 = c_m^2$.

Let $|S| = \alpha$ and $|\bar{S}| = \beta$. Then the above construction of S and $\bar{S}$ leads to the following inequality

$$\alpha c_m^2 \leq \beta c_M^1 + c_m^2. \tag{4}$$

Note that the term c_m^2 appears at right hand of the above inequality because there might be some budget left out after constructing the minimum cardinality feasible set $\bar{S}$ and it is at most c_m^2. Now equation (4) implies that

$$\Rightarrow \frac{\alpha}{\beta} \leq \frac{c_M^1}{c_m^2} + \frac{1}{\beta}. \tag{5}$$

Note that $\beta \geq \frac{B}{c_M^1}$. This fact and expression (5) imply that

$$\Rightarrow \frac{\alpha}{\beta} \leq \frac{c_M^1}{c_m^2} + \frac{c_M^1}{B} \quad \Rightarrow \frac{|S|}{|\bar{S}|} \leq c_M^1\left(\frac{1}{c_m^2} + \frac{1}{B}\right). \tag{6}$$

On similar lines as above, we can also deal with the remaining three cases: (a) $c_M^1 > c_M^2$ and $c_m^1 \leq c_m^2$; (b) $c_M^1 \leq c_M^2$ and $c_m^1 > c_m^2$; and (c) $c_M^1 \leq c_M^2$ and $c_m^1 \leq c_m^2$. This completes the proof of the lemma. ■

Theorem 1. *The proposed greedy algorithm determines the initial seeds for the B-IMCP problem that is at least* $\frac{Bc_m}{B(c_M+c_m)+c_M c_m}$ *times the optimum solution, where* $c_M = max(c_M^1, c_M^2)$ *and* $c_m = min(c_m^1, c_m^2)$*.*

Proof. It is known that the greedy hill climbing algorithm gives a $\frac{1}{(p+1)}$ approximation of submodular function maximization for any p-system (Calinescu, Chekuri, Pal, and Vondrak (2007)). The proof of this theorem follows immediately as a consequence of this fact and Lemma 3. ■

In the view of the above results, we now present a few important observations as follows: (i) The approximation guarantee of the proposed greedy algorithm is independent of the number of products in P_1 and P_2. Also the approximation guarantee only depends on the budget and the minimum and maximum costs for providing free samples of the products in P_1 and P_2. (ii) If the cost of each product in P_1 and P_2 is 1, then $c_M = c_m = 1$. This implies that the greedy approximation algorithm returns a solution to the B-IMCP problem that is at least $\frac{B}{2B+1}$ times that of the optimum solution.

5 Heuristics for the B-IMCP Problem

We observe that the proposed greedy approximation algorithm runs slow even with data sets consisting of a few thousands of nodes (refer to Section 6 for more details). Though the design of a scalable heuristic for the B-IMCP problem is not the main focus of this paper, we here outline three heuristics for the proposed problem and we refer to the full version of this paper [22] for complete details about these heuristics.

(a) *Maximum Influence Heuristic (MIH):* The main idea behind this heuristic is motivated by Aggarwal *et al.* [23] and Chen *et al.* [8]. We now briefly present the main steps involved in the maximum influence heuristic as follows: (i) For each node $i \in V$, determine its influence spread; (ii) Sort the nodes in the network in non-increasing order of their influence spread values; and (iii) Pick nodes one by one as per the above sorted sequence and choose them as the initial seeds for appropriate products based on the ideas similar to that in Algorithm 1.

(b) *Maximum Degree Heuristic:* Following this heuristic, we first determine the nodes with high degree and then use steps similar to those presented in Algorithm 1 to construct the initial seed set.

(c) *Random Heuristic:* Following this heuristic, we select nodes uniformly at random to construct the initial seed set for the B-IMCP problem.

6 Experimental Evaluation

The goal of this section is to present experimental evaluation of the algorithms for the B-IMCP problem. We compare and contrast the performance of the proposed approximation algorithm, maximum influence heuristic, maximum degree heuristic and random heuristic. Throughout this section, we use the following acronyms to represent various algorithms: (i) *GA* to represent the proposed greedy approximation algorithm (i.e., Algorithm 1), (ii) *MIH* to represent the maximum influence heuristic, (iii) *MDH* to represent the maximum degree heuristic, and (iv) *Random* to represent the random heuristic. All the experiments are executed on a desktop computer with (i) Intel(R) Core (TM) i7 CPU (1.60 GHz speed) and 3.05 GB of RAM, and (ii) 32-bit Windows XP operating system. Each experimental result is taken as the average over $R = 1000$ repetitions of the same experiment. Further, we note that all the algorithms are implemented using JAVA.

6.1 Description of the Data Sets

Here we briefly describe the social network data sets that we use in our experiments.

WikiVote: This network data set contains all the users and discussion from the inception of Wikipedia till January 2008. Nodes in the network represent wikipedia users and a directed edge from node i to node j represents that user i voted on user j. This data set contains 7115 nodes and 103689 edges [24].

High Energy Physics (HEP): This is a weighted network of co-authorship between scientists posting preprints on the High-Energy Theory E-Print Archive between Jan 1, 1995 and December 31, 1999. This is compiled by Newman [25]. This network has 10748 nodes and 52992 edges.

Epinions: This is a who-trust-whom online social network of a general consumer review site Epinions.com. This data set consists of 75879 nodes and 508837 edges [26].

Telecom Call Data: This data set contains all the details pertaining to a call such as the time, duration, origin, and destination of the phone call. This data is collected from one of the largest mobile operators in a emerging market. We construct a graph from this data using the approach proposed in Nanavati *et al.* [27]. This data set consists of 354700 nodes and 368175 edges.

A summary of all the data sets described above is given in Table 6.1.

Table 1. Summary of the data sets used in the experiments

Data Set	Number of Nodes	Number of Edges
WikiVote	7115	103689
HEP	10748	52992
Epinions	75879	508837
Telecom Call Data	354700	368175

6.2 Experimental Setup

We follow the proposed LT-CP model of information diffusion. Given a weighted and directed social graph $G = (V, E)$ with a probability/weight w_{ij} for each edge (i, j) in E, we normalize these probabilities/weights as follows. Assume that node $i \in V$ has directed edges coming from nodes $j_1, j_2, \ldots, j_x$ with weights $q_{j_1 i}, q_{j_2 i}, \ldots, q_{j_x i}$ respectively. These weights represent the extent the neighbors of node i influence node i. Now let $q = q_{j_1 i} + q_{j_2 i} + \ldots + q_{j_x i}$; then the probability that node j_1 influence node i is given by $w_{j_1 i} = \frac{q_{j_1 i}}{q}$, the probability that node j_2 influence node i is given by $w_{j_2 i} = \frac{q_{j_2 i}}{q}$, and so on. Thus note that $w_{j_1 i} + w_{j_2 i} + \ldots + w_{j_x i} \leq 1$ and this setting coincides with the proposed model in Section 3.1.

We have carried out the experiments with several configurations of the parameters and we obtained similar results for each of these configurations. In the interest of space, we in particular present the results for the following configuration of the parameters. We consider two products each in P_1 and P_2 respectively; i.e. $|P_1| = |P_2| = t = 2$. Also we consider that $B = 100$, $c_1^1 = 5$, $c_2^1 = 4.5$, $b_1^1 = 7$, $b_2^1 = 6.5$, $c_1^2 = 3$, $c_2^2 = 2.5$, $b_1^2 = 5$, $b_2^2 = 4.5$.

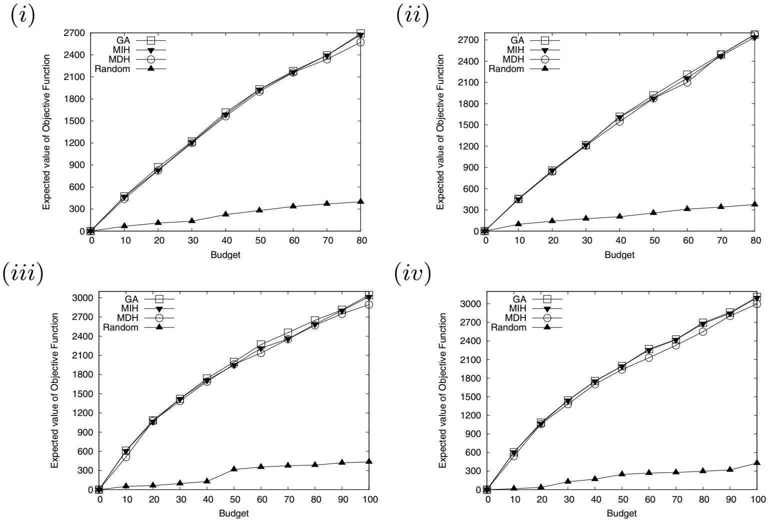


Fig. 1. Performance comparison of GA, MIH, MDH, and Random when the first variant of cross-sell is considered and (i) Dataset is HEP and cross-sell threshold is $[0, 0.5]$, (ii) Dataset is HEP and cross-sell threshold is $[0, 0.2]$, (iii) Dataset is WikiVote and cross-sell threshold is $[0, 0.5]$, and (vi) Dataset is WikiVote and cross-sell threshold is $[0, 0.2]$

We also work with two intervals for the cross-sell thresholds by setting $a = 0.2$ and $a = 0.5$ (refer to Section 3.1). This implies that the cross-sell thresholds come from two types of intervals, namely $[0, 0.2]$ and $[0, 0.5]$.

6.3 Experimental Results

We would like to compute the value of the objective function for the B-IMCP problem by varying the budget level using the four algorithms, namely GA, MIH, MDH, and Random. The experimental results in this setting are shown in Figure 1. These graph plots are obtained using HEP and WikiVote data sets and when the cross-sell thresholds come from the intervals [0,0.5] and [0,0.2] respectively. From all these graph plots, it is clear that the performance of MIH and MDH is almost same as that of GA. However, note that the performance of Random is very poor compared to that of GA.

Experiments with Large Network Data Sets. In this section, we focus on the running time of GA. Table 2 shows the running times of GA and MIH on HEP and WikiVote data sets. Clearly, the running time of GA is slower than MIH about 20 times.

Table 2. Running Times of GA and MIH on HEP and WikiVote Datasets

Data Set	Running Time of GA (in Sec.)	Running Time of MIH (in Sec.)
HEP	66563	2590
WikiVote	148736	7200

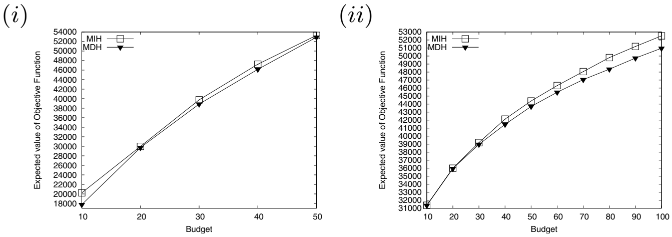


Fig. 2. Performance Comparison of MIH and MDH on Two Large Network Data Sets, namely (i) Epinions and (ii) Telecom Call Detail Records Data Sets

We now present experimental results with large network data sets using MIH and MDH. Figure 2 shows the budget versus the expected value of the objective function curves for MIH and MDH using Epinions and Telecom data sets, when the first variant of cross-sell is used and cross-sell thresholds for nodes come from the interval $(0, 0.5)$. It is immediate to see that the performance is MIH is superior than that of MDH on these two data sets.

7 Conclusions and Future Work

In this paper, we introduced a generalized version of the influence maximization problem by simultaneously considering three aspects such as cross-sell phenomenon, product specific costs and benefits and budget constraints. We proposed a simple greedy algorithm to address this generalized version of the influence maximization problem. There are several ways to extend this work in this paper. First, it is interesting to examine other types of possibility of the cross-sell among the products. Second, we considered an onto function to represent the cross-sell relationships in this paper. However, it is important to work with other types of functions to represent the cross-sell relationships while retaining the properties of the diffusion model such as monotonicity and submodularity.

References

1. Kempe, D., Kleinberg, J.M., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the 9th SIGKDD, pp. 137–146 (2003)
2. Rogers, E.M.: Diffusion of Innovations. Free Press, New York (1995)
3. Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press, Cambridge (2010)
4. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge University Press (1994)
5. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proceedings of the 7th SIGKDD, pp. 57–66 (2001)
6. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: Proceedings of the 8th SIGKDD, pp. 61–70 (2002)
7. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: SIGKDD, pp. 420–429 (2007)

8. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD, pp. 937–944 (2009)
9. Kimura, M., Saito, K.: Tractable Models for Information Diffusion in Social Networks. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 259–271. Springer, Heidelberg (2006)
10. Mathioudakis, M., Bonchi, F., Castillo, C., Gionis, A., Ukkonen, A.: Sparsification of influence networks. In: Proceedings of the 17th SIGKDD (2011)
11. Even-Dar, E., Shapira, A.: A Note on Maximizing the Spread of Influence in Social Networks. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 281–286. Springer, Heidelberg (2007)
12. Datta, S., Majumder, A., Shrivastava, N.: Viral marketing for multiple products. In: Proceedings of IEEE ICDM, pp. 118–127 (2010)
13. Bharathi, S., Kempe, D., Salek, M.: Competitive Influence Maximization in Social Networks. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 306–311. Springer, Heidelberg (2007)
14. Borodin, A., Filmus, Y., Oren, J.: Threshold Models for Competitive Influence in Social Networks. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 539–550. Springer, Heidelberg (2010)
15. Carnes, T., Nagarajan, C., Wild, S.M., van Zuylen, A.: Maximizing influence in a competitive social network: a follower's perspective. In: Proceedings of the 9th International Conference on Electronic Commerce (ICEC), pp. 351–360 (2007)
16. Granovetter, M.: Threshold models of collective behavior. American Journal of Sociology 83, 1420–1443 (1978)
17. Schelling, T.C.: Micromotives and Macrobehavior. W.W. Norton and Company (1978)
18. Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a submodular set function subject to a matroid constraint. In: Proceedings of IPCO, pp. 182–196 (2007)
19. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions-1. Mathematical Programming 14(1), 265–294 (1978)
20. Goldenberg, J., Libai, B., Muller, E.: Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters 12(3), 211–223 (2001)
21. Ramasuri, N., Narahari, Y.: A shapley value based approach to discover influential nodes in social networks. IEEE Transactions on Automation Science and Engineering 8(1), 130–147 (2011)
22. Ramasuri, N., Nanavati, A.: Viral marketing with competitive and complementary products through social networks. Technical report, IBM Research, India (2012), http://lcm.csa.iisc.ernet.in/nrsuri/Cross-Sell-Suri-Amit.pdf
23. Aggarwal, C.C., Khan, A., Yan, X.: On flow authority discovery in social networks. In: Proceedings of SIAM Conference on Data Mining (SDM), pp. 522–533 (2011)
24. Leskovec, J., Huttenlocher, D., Kleinberg, J.M.: Signed networks in social media. In: Proceedings of the 28th ACM SIGCHI Conference on Human Factors in Computing Systems (CHI), pp. 1361–1370 (2010)
25. Newman, M.E.J.: The structure of scientific collaboration networks. Proceedings of the National Academy of Sciences (PNAS) 98, 404–409 (2009)
26. Richardson, M., Agrawal, R., Domingos, P.: Trust Management for the Semantic Web. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 351–368. Springer, Heidelberg (2003)
27. Nanavati, A.A., Singh, R., Chakraborty, D., Dasgupta, K., Mukherjea, S., Das, G., Gurumurthy, S., Joshi, A.: Analyzing the structure and evolution of massive telecom graphs. IEEE Transactions on Knowledge Discovery and Data Engineering 20(5), 703–718 (2008)

Which Topic Will You Follow?[*]

Deqing Yang[1], Yanghua Xiao[1,**], Bo Xu[1], Hanghang Tong[2], Wei Wang[1],
and Sheng Huang[3]

[1] School of Computer Science, Fudan University, Shanghai 200433, P.R. China
{yangdeqing,shawyh,xubo,weiwang1}@fudan.edu.cn
[2] IBM T.J. Watson Research Center, USA
htong@us.ibm.com
[3] IBM China Research Lab, P.R.China
huangssh@cn.ibm.com

Abstract. Who are the most appropriate candidates to receive a call-for-paper or call-for-participation? What session topics should we propose for a conference of next year? To answer these questions, we need to precisely predict research topics of authors. In this paper, we build a MLR (Multiple Logistic Regression) model to predict the topic-following behavior of an author. By empirical studies, we find that social influence and homophily are two fundamental driving forces of topic diffusion in SCN (Scientific Collaboration Network). Hence, we build the model upon the explanatory variables representing above two driving forces. Extensive experimental results show that our model can consistently achieves good predicting performance. Such results are independent of the tested topics and significantly better than that of state-of-the-art competitor.

Keywords: topic-following, social influence, homophily, SCN.

1 Introduction

User behavior understanding and prediction are important tasks in social computing. One of the typical tasks is to underhand the author behavior from the public publication records and one of the most interesting author behaviors is topic-following. In general, among all possible topics, an author may select one or several as his future research topics due to his limited time and efforts. Then, a problem will naturally arise: *Can we predict the topic of the next paper for an author?* More specifically, given the historical publications of an author, can we predict the most possible topic of his next papers? In this paper, we answer this question with a positive answer by successfully modeling the topic-following behavior of authors.

[*] This work was supported by NSFC under grant Nos 61003001, 61033010, 60673133 and 60703093; Specialized Research Fund for the Doctoral Program of Higher Education No. 20100071120032; and partly supported by Zhejiang Provincial NSFC (LY12F02012). The fourth author is support in part by DAPRA under SMISC Program Agreement No. W911NF-12-C-0028 and by the U.S. Army Research Laboratory under Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the offcial policies, either expressed or implied, of DARPA, ARL, or the U.S. Government.

[**] Corresponding author.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 597–612, 2012.

One may directly use the historical topics of an author to predict the topic of his/her next papers. However, such information in general is insufficient for acceptable accuracy of prediction. Because an author's topic-following behavior is subject to many other factors, such as the influence from his/her collaborators, the current popular topics, historical topics etc. These factors are usually mixed together to affect an author's topic-following behavior.

In this paper, by empirical studies, we found that the topic-diffusion on the co-author networks has significant influence on authors' topic-following behavior. Hence, our basic idea is first constructing a scientific collaboration network, and then model the users' topic-following behaviors by explanatory variables observed from the topic-diffusion among authors in the network.

1.1 Applications

Our research is driven by the following real applications:

1. *Call for participation or paper submission*. When a workshop for a certain topic is announced, delivering the call-for-paper or call-for-participation to the most appropriate candidates who are interested in the topic is critical for the success of the workshop.
2. *Proposal of session topic*. Suppose we need to organize a conference of the next year. What topics should be proposed as sessions of the conference to attract as many attendees as possible? If an accurate topic-following model is available, we can easily summarize the amount of potential audience for sessions of different topics.

The model can also find more applications, such as advertisement, friend recommendation etc. For example, in online social networks, by identifying the topics of posts or comments produced by users, we can deliver advertisements of the topic to potential users who are recognized by the topic-following model [1]. In addition, we can recommend the users who will follow the same topic as the friends of the objective users [2].

1.2 Topic Diffusion

Topic diffusion in *Scientific Collaboration Network* (SCN) is one of important processes that may influence the topic-following behavior of an author. SCN is a co-author network, in which each vertex is an author and each edge represents a co-author relationship. Intuitively, if a topic is diffused to an author from many of his coauthors in the SCN, it is of high probability that he will adopt the topic in his future publications. In this paper, *we build our topic-following model based on the topic-diffusion principles in SCN.*

Generally speaking, there are two typical ingredients which impact information diffusion among individuals in a social network: *social influence* and *homophily* [3, 4]:

1. Social influence means that an individual tends to adopt behaviors of his neighbors or friends.

2. Homophily is the tendency of individuals to choose friends with similar characteristics [3, 5].

Social influence depends on the structure of the social network. In contrast, homophily focuses on the attribute similarity among individuals, in other words, it does not matter whether they are connected to each other. These two factors have been widely investigated as the underlying mechanisms accounting for linkage formation in large social networks [5].

1.3 Challenges and Contributions

Thus, we first need to understand the effect of *social influence* and *homophily* on topic diffusion in SCN before we can precisely model authors' topic-following behavior. Unfortunately, most previous research work on SCN mainly focus on macroscopic structure of the whole network [6, 7], collaboration pattern [8] or community evolving [9], leaving topic diffusion in SCN rarely investigated. Many findings about information propagation[1] on other social networks have been discovered, but the diffusion laws observed on these networks in general do not necessarily hold in SCN any more.

Hence, the purpose of this paper is two-fold. First, understanding the effects of *social influence* and *homophily* on topic diffusion in SCN. Second, developing an effective topic-following model based on the above findings. However, there still exist many challenges that remain unsolved.

- First, it is difficult to distinguish impacts of social influence and homophily from each other. Because they are often mixed together [10] to affect topic diffusion. Furthermore, quantifying their impacts on research topic-following is subjective.
- Second, it is hard to accurately define topics for papers due to the uncertainty and multiplicity of topic identification. Since topic-following behaviors of authors are topic-sensitive, precisely defining the topic of a paper is critical for the model's performance.
- Third, sample sparseness poses a great challenge. Many scientists have quite small number of papers and many topics have only a few papers, which generally bring great challenges to accurately predict the authors' topic-following behavior.

In this paper, we address above challenges and make the following contributions:

- First, we uncover the effects of social influence and homophily on topic diffusion in SCN by extensive empirical studies.
- Second, we propose a *Multiple Logistic Regression* (MLR) model based on the empirical results to predict topic-following of authors.
- Third, we conduct extensive experiments with comparison to the state-of-the-art baseline to show the advantage of our proposed model in prediction performance.

Although our model is proposed for SCN, it can also be used in other social settings, for example, predicting buyer behavior in e-commerce, topic prediction in microblogging etc.

[1] In the following texts, *information propagation* or *information spreading* may also be used interchangeably with *information diffusion*.

1.4 Organization

The rest of this paper is organized as follows. Sect. 2 is a brief review of related work. We introduce the basic concepts and try to identify the effects of social influence and homophily on topic diffusion in SCN in Sect. 3. In Sect. 4, we present empirical results about driving forces of topic propagation in SCN. Based on the findings in empirical analysis, in Sect. 5, we propose a MLR model to predict topic-following of authors with the comparisons to the baseline approach. At last, we conclude our work in Sect. 6.

2 Related Work

We review the related works from the following three aspects: *information diffusion*, *scientific collaboration network*, and *user behavior modeling*.

Information diffusion. Topic diffusion can be regarded as a special case of information/idea propagation on social networks, which has already been studied in sociology, economics, psychology and epidemiology [11–13]. Many research work of information diffusion focused on concrete object propagation on online Web media, such as article diffusion on Wikipedia [4], picture diffusion on Flickr [3], post diffusion on Blogsphere [14, 15] and event diffusion on Twitter [16], but for topic addressed in this paper, it is rarely explored in terms of information diffusion on social networks. Although D.Gruhl *et al.* studied topic diffusion [15]. But their focus is the social network in Blogsphere other than SCN. Research topics of SCN have also been investigated in [17, 18]. But they focused on detecting topic evolution and transition over time. Social influence and homophily have been regarded as two major causal ingredients [3, 10, 4] of information diffusion on social networks. It is widely established that it is social influence and homophily as well as their interactions that determine the linkage formation of individuals [19, 13] or interplays between two individuals in social networks [20]. It is a traditional belief that social influence accounts for the information diffusion on typical social networks [11, 14]. However, recent study in sociology argues that homophily also plays an important role for individuals to follow others' idea or adopt others' behavior [21]. As a result, some literatures [22, 4] studied the cumulative effects of social influence and homophily on diffusion cascading. However, to the best of our knowledge, the effects of social influence and homophily on research topic diffusion in SCN have rarely been reported.

Scientific collaboration network As a typical social network, SCN was systematically investigated by Newman *et al.* [6, 8]. But they only focused on the structural properties of the network without exploring topic diffusion. The SCN constructed in [23] is identical to the one used in this paper, but it studied the evolution of collaboration between individuals instead of research topic diffusion. Tang *et al.* [24] also investigated topic-level social influence on SCN, but they did not take homophily's influence into account.

User behavior modeling Information spreading can be considered as one kind of user behavior. Many user behavior models have been proposed in previous studies. For example, some works [25, 26] modeled retweet patterns of users in Twitter. Others [27, 3] modeled the user interaction pattern in Flickr and MySpace. All these works did not model topic-following behavior in SCN.

3 Preliminaries

In this section, we first review the preliminary concepts about topic diffusion in SCN. Then we propose our solution to quantify the influence of social influence and homophily on topic diffusion in SCN.

3.1 Basic Concepts

We first formalize SCN and explain the rationality to use SCN, then formalize the concept of author's topic-following behavior in SCN.

Scientific Collaboration Network (SCN) is a co-author network. We give the formal definition of SCN in Definition 1. Notice that SCN is evolving over time, the snapshot of SCN at time t is denoted by G_t. Its vertex set and edge set are denoted by V_t and E_t, respectively. G_t encodes all coauthor relationships till time t. In other words, if $t_1 \leq t_2$, G_{t_1} is a subgraph of G_{t_2}, i.e., $V_{t_1} \subseteq V_{t_2}, E_{t_1} \subseteq E_{t_2}$. And for an edge $e_{u,v} \in E_{t_1}$, we have $w_{t_1}(e_{u,v}) \leq w_{t_2}(e_{u,v})$.

Definition 1 (SCN). *A Scientific Collaboration Network is an undirected, edge-weighted graph $G = (V, E, w)$, where node set V represents authors, edge set E represents coauthor relationships, and $w : E \rightarrow \mathbb{N}$ is the weight function of edges. For each edge $e_{u,v} \in E$, $w(e_{u,v})$ is defined as the number of papers that u and v have ever coauthored.*

Rationality to Use SCN. In this paper, we mainly focus on SCN constructed from DBLP data set. The reason is two-fold.

- First, it is a good approximation of social networks in real life since most coauthors are acquainted to each other. SCN shares many generic properties of a social network. Most principles guiding the users' behavior on social networks still hold true.
- Second, SCN extracted from DBLP contains enough clean information. DBLP contains plenty of computer science publication records, each of which includes title, author list, venue information and publishing year. These information allows us to explore the topic-following behavior of authors. DBLP dataset is cleaned before its publication. Some noise in the data, such as name ambiguity, has been preprocessed. Thus, the extracted SCN is free of such noise.

Topic Diffusion. Given a topic s, we say an author u followed s if u has published at least one paper of s. Moreover, the set of authors who published at least one paper of topic s in year t is denoted as U_t^s whose size is $|U_t^s|$. Similarly, authors who published papers of s up to year t are denoted by $U_{\leq t}^s$. Then, *the popularity of topic s in year t* can be measured by $|U_t^s|$. The *diffusion of topic s* is a dynamic process which can be observed from the evolution of $|U_t^s|$ along time t. DBLP only records the year when a paper was published, thereby the unit of one time step is defined as one year when we study temporal properties of topic diffusion.

Dataset Description and Topic Extraction. We select the papers published up to year 2011 from the seven major categories[2], e.g., *database*, *data mining*, *World Wide Web*, to

[2] http://academic.reserach.microsoft.com

construct SCN. The resulting SCN contains 193,194 authors and 557,916 co-authoring relationships. Identifying the topic of each paper is a preliminary step for the study of topic diffusion. We select 25 representative topics, such as *Query Processing, Privacy and Security, and Social Networks, etc*. Then, we build a SVM [28] classifier trained on a manually-labeled dataset to classify each paper into the 25 topics.

3.2 Social Influence and Homophily

In this subsection, we present our solution to evaluate the effects of social influence and homophily on topic propagation in SCN. We first show that it is intractable to precisely distinguish them from each other. Hence, we turn to a qualitative way to evaluate the two factors' effects. In general, it was well established that the more neighbors adopting an idea, the more possible himself will follow the idea. Thus, we evaluate the effect of social influence by the number of neighbors who have adopted a topic before. In DBLP, the *topic similarity* is a good indicator of the homophily between two authors. Thus, we use topic similarity to evaluate the effect of homophily.

Intractability. In general, it is intractable to precisely distinguish the effects of social influence and homophily from each other [10, 3]. We illustrate this by Figure 1. In the graph, a dark node represents an author who has followed a certain topic (say s). In G_{t-1}, only one author a has ever published a paper of topic s. Then in G_t, author d, h, f, g also adopt the topic. Since d and h are the direct neighbors of a, we can assume that they are infected by a's influence through social ties between them. However, we can not exclude the possibility that the topic-following behavior of d and h is due to their own interests on the topic. f and g have no direct links to a. They are linked to a only by two-step paths. Hence, we may assume that their topic-following behaviors are mainly due to homophily since in general social influence through indirect links is weak. However, it is also possible that e learned about topic s from a and then recommended it to his neighbors f and g. Hence, it is hard to precisely quantify the effect of social influence and homophily.

Social Influence in SCN. Social influence refers to the process in which interactions with others cause individuals to conform, e.g., people change their attitudes to be more similar to their friends [22, 4]. In the context of topic diffusion in SCN, social influence can be characterized as the tendency of an author adopting the same topic as his neighbors. In general, the more neighbors infected by the topic, the more tendency he will adopt the same topic of his neighbors. Thus, the effect of social influence can be directly evaluated by the the number of neighbors who have published papers of a certain topic [10]. In other words, if an author followed a topic at t and a significant number of his neighbors (i.e., coauthors) had ever published papers of this topic before year t, it would be of high confidence that the author's topic-following behavior is affected by social influence.

Homophily in SCN. In our study, we use topic similarity among authors to approximate homophily in SCN. Homophily can be regarded as demographic, technological, behavioral, and biological similarities of individuals [10, 5]. It is intractable to precisely define homophily in SCN due to the limited information available from DBLP dataset. One good approximation in SCN is *topic similarity*. We use the *history topic vector* to represent an author's research interests, which can be formally defined as $\boldsymbol{u} = [n_1, n_2, ..., n_{25}] \in \mathbb{N}^{25}$, here each n_i is the number of the author u's papers belonging to i-th topic. Then, the topic similarity between author u and v can be given as follows:

Definition 2 (Topic Similarity). *Given two authors u and v, the topic similarity of author u and v is defined as,*

$$sim(u, v) = cosine(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\parallel \boldsymbol{u} \parallel \parallel \boldsymbol{v} \parallel} \tag{1}$$

Note that $\boldsymbol{u}$ and $sim(u, v)$ are time-dependent variables, which are calculated within a time window. Recall that $|U_t^s|$ varies as time elapses. By summarization, we find that most topics' $|U_t^s|$s keep above 80% of peak value only for three years, indicating most scholars retain their interests of one topic for about three years. Hence, we will count n_i according to the papers published in a three-year time window $[t-3, t-1]$ when computing $\boldsymbol{u}$ at time t.

Further we define the topic similarity between one author u and a group of authors U. Similarly, we first define a history topic vector for U as $\boldsymbol{\mathcal{U}} = [N_1, N_2, ..., N_{25}]$, where N_i is the total number of papers of i-th topic composed by any one in U, then we have $sim(u, U) = cosine(\boldsymbol{u}, \boldsymbol{\mathcal{U}})$. $\boldsymbol{\mathcal{U}}$ is also calculated within three-year window.

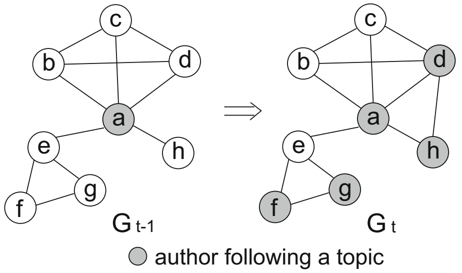


Fig. 1. Illustration of topic diffusion. Social influence and homophily are mixed together to affect topic diffusion.

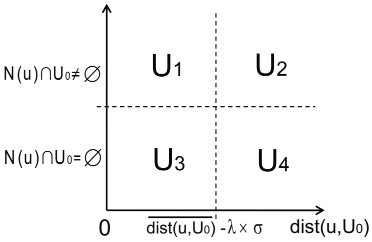


Fig. 2. The division of $\bar{U}_0$

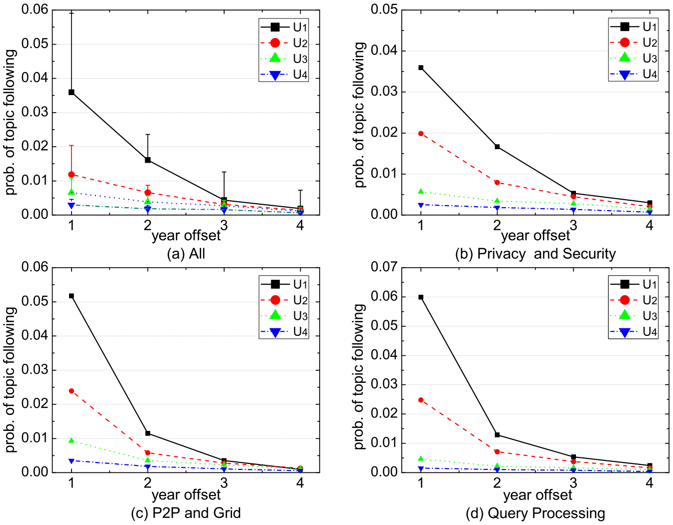
4 Empirical Study

In this section, we present the empirical study results. Our purpose of empirical study is two-fold. First, in Sec 4.1 we show that social influence and homophile are two fundamental driving forces of topic diffusion in SCN. Second, we reveal the way that social influence affects topic diffusion in Sec 4.2.

4.1 Driving Forces of Topic Diffusion

We first give the detail of our experiment design, then give the results.

Experiment Design. Let $U_0 = U^s_{\leq t_0}$, i.e., the set of authors who have published papers of topic s till t_0. We will focus on $\bar{U}_0 = V_{t_0} - U_0$, i.e., those who have not published any papers of topic s till t_0. $\bar{U}_0$ can be divided into four disjoint subsets U_1, U_2, U_3 and U_4 [3], according to two conditions. This division of $\bar{U}_0$ is illustrated in Figure 2, where $N(u)$ is the neighbor set of u.



(a) All

(b) Privacy and Security

(c) P2P and Grid

(d) Query Processing

Fig. 3. Evolution of driving forces of individuals' topic-following. Both social influence and homophily are effective on topic diffusion in SCN. And both of them decay in an exponential way.

The first condition is whether there exist neighbors that belong to U_0. If exist, this author may publish a paper of topic s due to social influence [19, 3]. The second is the discrepancy of an author's topic vector to U_0's. Specifically, for each author $u \in \bar{U}_0$, we can calculate $dist(u, U_0) = 1 - sim(u, U_0)$, i.e., the distance between topic vectors of u and U_0. Then, we compute the standard deviation σ for $\{dist(u, U_0) | u \in \bar{U}_0\}$. We further set a threshold $\tau = \overline{dist(u, U_0)} - \lambda \times \sigma$, where $\overline{dist(u, U_0)}$ is the mean value and $0 < \lambda < 1$ is a tuning parameter. Any author $u \in \bar{U}_0$ with $dist(u, U_0) \leq \tau$ will be identified as the one whose topic vector is sufficiently similar to U_0. These authors may publish a paper of topic s after t_0 driven mainly by homophily [3].

Accordingly, if there are authors in U_i $(1 \leq i \leq 4)$ publishing a paper of topic s after t_0, those in U_1 may be affected by social influence as well as homophily; those in U_2 are affected merely by social influence; those in U_3 may be affected merely by homophily. While U_4 represents the remaining authors who are not influenced by the two forces with high probability.

For each U_i $(1 \leq i \leq 4)$, we count the number of authors who publish the paper of topic s after t_0 for each s. Then, we calculate the proportion of authors within each U_i that follow the topic. This proportion can be regarded as the probability that an author within each group will follow the topic. Each proportion is normalized over all

[3] We may also use $U_i(s)$ to denote each U_i when topic s needs to be specified explicitly.

topics. For example, the proportion of authors within U_1 over all topics is normalized as: $\frac{\sum_s |U_1'(s)|}{\sum_s |U_1(s)|}$, where $U_1(s) \subset V_{t_0} - U^s_{\leq t_0}$ and $U_1'(s)$ is the set of authors in $U_1(s)$ that followed topic s after t_0.

Results. The experimental results are shown in Fig. 3, where $t_0 \in [2005, 2007]$ and $\lambda = 0.8$. Fig. 3(a) shows that authors exhibiting more topic similarity to U_0 or having more neighbors in U_0 are more probable to follow the topic than those without these characteristics. We also can see that the cumulative effect of social influence and homophily on topic diffusion is more significant than either one of these forces. Moreover, it can be observed that the effects of social influence, homophily and their mixture are decaying in an exponential way as time elapses. Generally, three or four years later after t_0, minor effects can be observed (Similar results can be observed when we vary the year window to compute $dist(u, U_0)$). These facts indicate that social influence and homophily are generally time-sensitive. When we compare social influence to homophily, we find that social influence is more sensitive to time. These findings provide additional evidence for the time-sensitivity of social influence, which was first discovered in the study of product-adopting behavior [10].

All above findings are generally consistent with those found in specific topics, e.g., *Privacy and Security, P2P and Grid* and *Query Processing*, as shown in Fig. 3(b)∼(d). Different topics only show minor difference on the decaying speed.

4.2 Social Influence

In this subsection, we show that the number of infected neighbors and relationship strength have positive influence on topic diffusion.

Dependency on the Number of Infected Neighbors. It has been shown that the probability that an individual joins a group depends on the number of his friends in this group [19]. Then, *does an individual's topic-following behavior also depend on the number of his neighbors who have followed the topic before?* We get a *positive* result from the following studies.

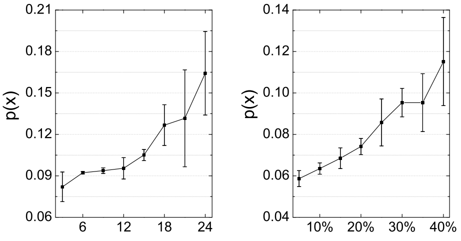
We first summarize the probability p with which an author follows his neighbor's research topic, as the function of the number or the proportion of his neighbors that have followed the topic. Let U_x be the set of authors that have x neighbors who have ever published papers of a given topic before. In U_x, some of them will follow the behavior of their neighbors to publish papers of the same topic. The set of such authors is denoted by U_x'. Thus, for each value of x, we can define $p(x)$ as:

$$p(x) = \frac{|U_x'|}{|U_x|} \tag{2}$$

$p(x)$ can be similarly defined when x is the proportion of neighbors who have ever published papers of a certain topic before.

The correlation between $p(x)$ and x is shown in Fig. 4. It is clear that for either case when x is the number (Fig. 4(a)) or proportion (Fig. 4(b)), $p(x)$ generally increases with x, strongly suggesting the probability that an author will follow a topic heavily depends on the number/proportion of his neighbors who have followed the topic.

All above results about neighbor's influence are consistent with classical diffusion theory. It was shown in [11] that innovation decision is made through a cost-benefit analysis where the major obstacle is uncertainty. Similarly, in topic diffusion, when more neighbors have followed a topic, other authors will be more certain about the benefit of following a certain topic, and consequently it is quite probable that an individual is persuaded to accept it.



(a) x as #

(b) x as proportion

Fig. 4. p(x) vs x, shows that an author's topic-following behavior depends on the number/proportion of his neighbors who have followed the topic.

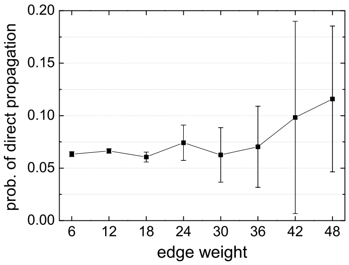


Fig. 5. Prob. of being infected vs edge weight. This figure shows that the strength of coauthoring is influential on the direct propagation from an author to his neighbors.

Dependency on Strength of Coauthoring. Recall that in SCN (Definition 1), each edge is assigned a weight indicating the number of coauthored papers. Thus, *whether the strength of the edge is influential on the direct propagation from an author to his neighbors?* The answer is *Yes* based on the following studies.

To answer the question, we summarize the correlation between the strength of coauthoring and the probability that a topic is propagated from an author to his neighbors. In general, more coauthored papers imply more common research interests, or other similarities between authors. Hence, it is expected that the probability of direct propagation is positively correlated to edge weight. The plot shown in Fig. 5 verifies our conjecture. In the figure, the *probability of direct propagation* is measured by *the proportion of edges on which direct propagation happens*, and is plotted as a function of edge weight. It is evident from the figure that direct propagation probability increases with the growth of edge strength. In other words, an individual is more likely to follow the research topics of his neighbors who have tighter relationships with him. This observation is consistent with our intuition that one person is likely to share his friend's interests or follow his friend's ideas.

5 Modeling Topic Diffusion in SCN

Based on the previous empirical results, in this section, we will propose a MLR model to predict the topic-following behavior of authors in SCN. Next, we will present the detail to build the model and evaluate predicting performance of the model.

5.1 Model Selection

Recall that for the authors who have not yet published any papers of a given topic before t, we intend to accurately estimate the number of them that will/or not follow the topic in t and future. This problem can be casted as a typical binary classification problem. *Logistic Regression Model* is one of the most widely used binary classifiers, which has been widely used in applications of medicine, genetics, business and etc. In this paper, since the behavior of individuals' topic-following is driven by more than one force, we adopt *Multiple Logistic Regression* (MLR for short)[29] to predict topic diffusion in SCN.

In MLR, dependent variable Y is a binary response variable with only two possible values, 1 or 0, which respectively represents whether an author will or not follow a certain topic if he has not adopted it before. And the value of Y relies on the multiple explanatory variables x_i, each of which represents an influential factor that affects an author's topic-following behavior. Let $\pi(x) = P(Y = 1)$ be the probability that an author will follow a certain topic. In MLR model, a linear relationship is established between *logit* function of $\pi(x)$ (or log odds of $\pi(x)$) and p explanatory variables. The detailed model can be described as the following equation:

$$logit[\pi(x)] = \ln \frac{\pi(x)}{1-\pi(x)} = \alpha + \Sigma_{i=1}^{p}\beta_i x_i \tag{3}$$

By simple transformation, we can calculate $\pi(x)$ by the following equation:

$$\pi(x) = 1/(1 + e^{-(\alpha+\Sigma_{i=1}^{p}\beta_i x_i)}) \tag{4}$$

where we have $0 \leq \pi(x) \leq 1$. Both α and β_i are the parameters that can be estimated by training the model. Since MLR is used as a binary classifier, we still need a cutoff value (cv for short) to help us classify each author into two categories. The simplest rule to use cv for classification is: *if* $\pi(x) \geq cv$*,* $Y = 1$*; otherwise* $Y = 0$. Typically, $cv = 0.5$ is used.

5.2 Explanatory Variables

Previous empirical studies suggest two explanatory variables representing social influence and homophily, respectively, to model the probability of topic-following. As we can see in Fig. 3, topic-following behavior of an individual in SCN varies as time elapses. So the two explanatory variables are time-dependent and are always discussed w.r.t. year t.

For Social Influence. We have shown that the probability that an author follows a topic is positively correlated to the number of his neighbors who have already followed the topic, as well as the strength of social ties between them. Similar to belief propagation model on factor graph [30], we quantify social influence as follows. For an author u, the probability that u follows the topic s at year t can be given as:

$$F_{SI}(u,s,t) = \sum_{v\in N'(u)} \frac{w(e_{u,v})}{\sum_{v\in N'(u)} w(e_{u,v})} \times f(v,s,t-1) \tag{5}$$

where $N'(u)$ is the neighbors of u who have followed topic s before u, $w(e_{u,v})$ is the weight of edge $e_{u,v}$ and $f(v,s,t-1)$ quantifies the influence from u's neighbor v in $t-1$. The function $f(\cdot)$ can be precisely defined as,

$$f(v,s,t) = \delta F_{SI}(v,s,t) + \frac{n_t^s}{n_t} \tag{6}$$

where $0 < \delta < 1$ is a punishing parameter, n_t is the number of u's publications at t among which n_t^s papers belong to topic s.

In the definition of $f(v,s,t)$, $\delta F_{SI}(v,s,t)$ summarizes the influence inherited from v's direct neighbors and indirect neighbors. As we have discussed in the example of Fig. 1, indirect neighbors may also have potential influence on topic-following by some intermediate authors. However, generally such indirect social influence degrades in an exponential way as the propagation length increases [31]. Hence, we need δ (δ=0.5 in our experiments) to punish the influence from faraway neighbors. The ratio $\frac{n_t^s}{n_t}$ accounts for v's interest on topic s in year t.

The computation starts from $F_{SI}(u,s,t_0)$ for each author u with $t_0 = 2002$. The initial value is set as $\frac{n^s_{<t_0}}{n_{<t_0}}$. Then, the computation proceeds iteratively for each year raning from t_0+1 to t. As above, Equation 5 will produce large F_{SI} when an individual has many infected neighbors and retains strong relationships to these neighbors, which confirms the findings about driving effects of social influence.

For Homophily. Homophily indicates that an author u tends to follow the topic of those whose research topics are similar to himself. This factor can be captured by $F_{TS}(u,s,t)$, which can be directly defined as the topic similarity between an author u and the group of authors who have ever published paper of the same topic before year t:

$$F_{TS}(u,s,t) = sim(u, U^s_{<t}) \tag{7}$$

Finally, Equation 3 can be rewritten as,

$$logit[\pi(x)] = \alpha + \beta_1 F_{SI} + \beta_2 F_{TS} \tag{8}$$

We use maximum likelihood method to estimate all parameters, i.e., α and each β_i.

5.3 Sample Preparation

In this subsection, we introduce our sample selection for model training and testing.

Preparing the Samples. To build the MLR model, we collect publications in year $[2004, 2008]$ as the training data and the publications in year 2009 as the testing data. Note that we build MLR model for each topic since the parameters are topic sensitive.

Suppose now we need to generate samples for a certain topic s. In general, the topic-following behavior of authors who seldom publish papers in one year is subject to randomness, and hence their behaviors tend to be outliers. Therefore, we will only consider those authors who published *significant* number of papers in one year as the *valid* training samples. In our experiments, the threshold is set to *3* papers. Then, all valid authors will be collected for each year t in $[2004, 2008]$. Thus, each pair $< u, t >$ ($2004 \leq t \leq 2008$ and u is a valid sample) will be regarded as one training pair sample for topic s.

Now, for each pair sample $< u, t >$, we need to assign a value of 0 or 1 to the binary response variable Y. We process Y as follows: $Y = 1$ *if author* u *publishes at least one paper of topic* s *or other topics closely related to* s *during the three-year time window* $[t, t+2]$*; otherwise* $Y = 0$. The setup of three-year time window is due to the following two reasons. It generally takes one or two years (or even more) for an author to follow a certain topic. It also takes time for a topic to be diffused to more authors especially for a new topic.

Relaxing the Topics. In the computation of the response variable, topic s is relaxed to be itself or some other related topics, which is due to the fact that many topics are closely related to each other. For example, the topic *XML* is closely related to *Query Processing* since one of the core tasks in XML data management is XML query processing. As a result, many authors may publish papers containing more than one topic and usually change their research interests from one topic to another related one. Given an author u, we say a *topic transition* $s_1 \to s_2$ happens in year t_2 if u first published a paper of topic s_1 in year t_1 and then published a paper of topic s_2 in year t_2 such that $t_2 > t_1$. Based on it, we can define *topic transition probability* from s_1 to s_2 before year t as follows,

$$P(s_1 \to s_2, t) = \frac{\sum_{t_1 < t_2 < t} |U_{t_1}^{s_1} \cap U_{t_2}^{s_2}|}{|U_{<t}^{s_1}|} \tag{9}$$

Next we use the following equation to identify topic s' that is *closely related to* s:

$$\frac{P(s \to s', t)}{P(s \to s, t)} \geq \gamma \tag{10}$$

where γ is a threshold parameter defining the *topic closeness*. The rationale is that if a topic s transits to another topic s' with a high probability which is close to that of s transiting to itself, s and s' are supposed to be closely related to each other. In our experiment, we set γ as 0.65. We found that the $\gamma = 0.65$ can find intuitively appropriate related topics. For example, *P2P and Grid* is related to *Web Service and Semantics*, *Frequency Mining* is related to *Classification and Learning*.

Balanced Sampling. We found that the training samples are imbalanced distributed over two classes. For example, for topic *XML*, there are 9,127 negative samples ($Y = 0$) and 2,517 positive samples ($Y = 1$). Traditional classification model aims to minimize the number of errors made during training under the assumption of balanced data distribution over classes. They are therefore not suitable for class-imbalanced data. Hence, we undersample negative samples [32] to ensure the balanced distribution of positive and negative samples.

5.4 Model Evaluation

In this section, we will evaluate the predicting performance of our model. For comparisons, the regression model proposed in [3] is also tested as the baseline. The baseline model also tried to predict the probability of an individual's topic-following action. But the model uses only one variable a, i.e., the number of already-active friends. The baseline model is formulated as

$$logit[\pi(x)] = \alpha + \beta ln(a + 1) \tag{11}$$

Clearly, the baseline approach only considers the effect of social influence.

We first justify the rationality of the selected explanatory variables. For topic *XML*, we give the parameters of MLR and the baseline model estimated by maximum likelihood method in Table 1. From the table, we can see that in MLR, all the predictors can explain the response variable (all estimated β_is are significant enough ($Sig. < 0.05$)), hence should be imported into MLR model as explanatory variables. Furthermore, we can see that F_{TS} is more influential to response variable than F_{SI} since β_2 as well as its $Wald$ is larger than β_1. Similar results can be obtained on other topics.

Table 1. Parameter estimation. *S.E.* is standard error of coefficients, *Wald* and *Sig.* are Wald Chi-square and P-value that test the null hypothesis of coefficient, respectively.

Model	Para.Name	Value	S.E.	Wald	Sig.
MLR	α	-1.620	0.064	631.5	0.00
	β_1	4.440	0.509	76.08	0.00
	β_2	9.566	0.346	763.6	0.00
baseline	α	-0.472	0.040	142.3	0.00
	β	0.808	0.044	338.4	0.00

Table 2. Predicting performance of MLR and the baseline model on *XML*, $\beta = 1.1$ in F_β computation

Metrics	MLR	baseline
recall/sens.	72.9%	70.3%
precision	57.9%	47.3%
F_β	65.3%	57.6%
specificity	65.4%	48.7%
accuracy	68.4%	57.2%

Predicting Performance of the Model. In general, the performance of a binary classifier can be measured by $sensitivity$, $specificity$, $precision$ and $accuracy$ [32], where $sensitivity$ (or $recall$) is the proportion of positive samples that are correctly predicted by the model, $specificity$ is the proportion of negative samples that are correctly predicted, $precision$ is the proportion of instances classified as positive that are really positive, and $accuracy$ is the proportion of samples that are correctly predicted either positive or negative. We give these metric results in Table 2 which shows that for all the tested accuracy indicators, MLR is prior to the baseline model. In some applications, for example, finding potential participants of a conference, we hope that more person who are really interested in a certain topic can be found. In other words, in these cases, improving *recall* and *precision* are more preferred. Hence, we also use F_β measure to evaluate the combined score of recall and precision [32].

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall} \tag{12}$$

We set $\beta = 1.1$ to favor recall a little.

Table 2 summarizes the prediction performance of MLR and the baseline model against test samples. We find that MLR outperforms its competitor for each metric. Specially, MLR outperforms the baseline model by about 20% with regard to accuracy, and by 13% with regard to F_β. MLR achieves almost 70% *accuracy* and F_β, which suggests that MLR is practically valuable in real applications.

We further give *accuracy* and F_β on each topic. As shown in Fig. 6 and Fig. 7, the advantage of MLR model over the baseline model can be consistently observed independent on all the tested topics. Fig. 8 further shows the ROC (receiver operating characteristic) curves [33] of MLR and the baseline model, where the area under MLR's ROC curve is 0.743 (area > 0.7 generally implies good predicting performance) suggesting our model is more effective to predict topic-following than the baseline (whose area is 0.638).

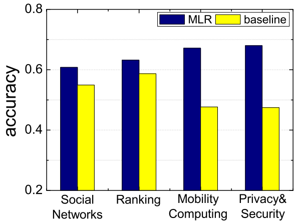


Fig. 6. Comparison of predicting accuracy

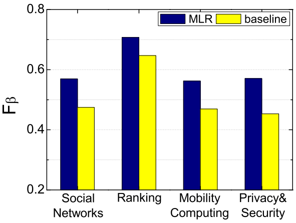


Fig. 7. Comparison of predicting F_β

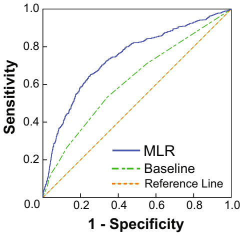


Fig. 8. ROC curves show that MLR is more effective than the baseline

6 Conclusion

Motivated by many real applications, such as call for participation or paper submission, we build a Multiple Logistic Regression model (MLR) to predict the topic that an author will adopt. We build the model upon our understanding about the topic diffusion in Scientific Collaboration Network (SCN). We find that social influence and homophily are mixted together to affect topic-following behavior of authors in SCN through empirical studies. We also uncover the characteristics that social influence affects topic diffusion. By extensive experimental studies, we show that our model can consistently achieves close to 70% accuracy and good F_β. Such results significantly outperform the state-of-the-art competitor model and can be applied in real applications.

References

1. Provost, F.J., Dalessandro, B., Hook, R., Zhang, X., Murray, A.: Audience selection for on-line brand advertising: Privacy-friendly social network targeting. In: Proc. of SIGKDD (2009)
2. Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., Leiser, N., Matias, Y., Merom, R.: Suggesting friends using the implicit social graph. In: Proc. of SIGKDD (2010)
3. Anagnostopoulos, A., Kumar, R., Mahdian, M.: Influence and correlation in social networks. In: Proc. of SIGKDD (2008)
4. Crandall, D., Cosley, D., Kleinberg, J., Huttenlocher, D., Suri, S.: Feedback effects between similarity and social influence in online communities. In: Proc. of SIGKDD (2008)
5. McPherson, M., Smith-Lovin, L., Cook, J.: Birds of a feather: Homophily in social networks. Annual Review of Sociology 27, 415–445 (2001)
6. Newman, M.E.J.: The structure of scientific collaboration networks. PNAS 98(2), 404–409 (2001)
7. Newman, M.E.J.: Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. Physical Review E 64(016132), 1–7 (2001)
8. Newman, M.E.J.: Coauthorship networks and patterns of scientific collaboration. PNAS 101, 5200–5205 (2004)
9. Wu, B., Zhao, F., Yang, S., Suo, L., Tian, H.: Characterizing the evolution of collaboration network. In: Proc. of SWSM (2009)
10. Aral, S., Muchnika, L., Sundararajan, A.: Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. PNAS 106, 21544–21549 (2009)

11. Rogers, E.: Diffusion of Innovations. Free Press (1995)
12. May, R.M., Lloyd, A.L.: Infection dynamics on scale-free networks. Physical Review E (2001)
13. Gomez-Rodriguez, M., Leskovec, J., Krause, A.: Inferring networks of diffusion and influence. In: Proc. of SIGKDD (2010)
14. Yang, J., Leskovec, J.: Modeling information diffusion in implicit networks. In: Proc. of ICDM (2010)
15. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. In: Proc. of SIGKDD (2004)
16. Lin, C.X., Zhao, B., Mei, Q., Han, J.: Pet: A statistical model for popular events tracking in social communities. In: Proc. of SIGKDD (2010)
17. Zhou, D., Ji, X., Zha, H., Giles, C.L.: Topic evolution and social interactions: How authors effect research. In: CIKM (2006)
18. He, Q., Chen, B., Pei, J., Qiu, B., Mitra, P., Giles, C.L.: Detecting topic evolution in scientific literature: How can citations help? In: Proc. of CIKM (2009)
19. Backstrom, L., Huttenlocher, D., Kleinberg, J.M., Lan, X.: Group formation in large social networks: Membership, growth, and evolution. In: Proc. of SIGKDD (2006)
20. Scholz, M.: Node similarity is the basic principle behind connectivity in complex networks. arXiv:1010.0803[physics.soc-ph] (2010)
21. Benjamin Golub, M.O.J.: How homophily affects diffusion and learning in networks. arXiv:0811.4013[physics.soc-ph] (2008)
22. Fond, T.L., Neville, J.: Randomization tests for distinguishing social influence and homophily effects. In: Proc. of WWW (2010)
23. Huang, J., Zhuang, Z., Li, J., Giles, C.L.: Collaboration over time: Characterizing and modeling network evolution. In: Proc. of WSDM (2008)
24. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: Proc. of SIGKDD (2009)
25. Peng, H.-K., Zhu, J., Piao, D., Yan, R., Zhang, J.Y.: Retweet modeling using conditional random fields. In: Proc. of ICDM Workshop (2011)
26. Macskassy, S.A., Michelson, M.: Why do people retweet? anti-homophily wins the day! In: Proc. of ICWSM (2011)
27. Choudhury, M.D., Sundaram, H., John, A., Seligmann, D.D.: Contextual prediction of communication flow in social networks. In: Proc. of WIC (2007)
28. Harris, D., Christopher, J.C., Linda, K., Smola Alexander, J., Vapnik, V.: Support vector regression machines. In: NIPS, pp. 155–161 (1996)
29. Agresti, A.: Categorical data analysis. Wiley, Berlin (2002)
30. Kschischang, F.R., Member, S., Frey, B.J., Andrea Loeliger, H.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47, 498–519 (2001)
31. Goetz, M., Leskovec, J., McGlohon, M., Faloutsos, C.: Information propagation and network evolution on the web. In: Proc. of CWSM (2009)
32. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann (2006)
33. Mark, G.: Receiver operating characteristic (roc) plots: Fundamental evaluation tool in clinical medicine. Clin. Chem. 30(4), 561–567 (1993)

Inferring Geographic Coincidence in Ephemeral Social Networks*

Honglei Zhuang[1], Alvin Chin[2], Sen Wu[1], Wei Wang[2]
Xia Wang[2], and Jie Tang[1]

[1] Department of Computer Science and Technology, Tsinghua University
[2] Nokia Research Center Beijing
zhl09@mails.tsinghua.edu.cn, jietang@tsinghua.edu.cn,
ronaldosen@gmail.com
{alvin.chin,wei.41.wang,xia.s.wang}@nokia.com

Abstract. We study users' behavioral patterns in ephemeral social networks, which are temporarily built based on events such as conferences. From the data distribution and social theory perspectives, we found several interesting patterns. For example, the duration of two random persons staying at the same place and at the same time obeys a two-stage power-law distribution. We develop a framework to infer the likelihood of two users to meet together, and we apply the framework to two mobile social networks: UbiComp and Reality. The former is formed by researchers attending UbiComp 2011 and the latter is a network of students published by MIT. On both networks, we validate the proposed predictive framework, which significantly improve the accuracy for predicting geographic coincidence by comparing with two baseline methods.

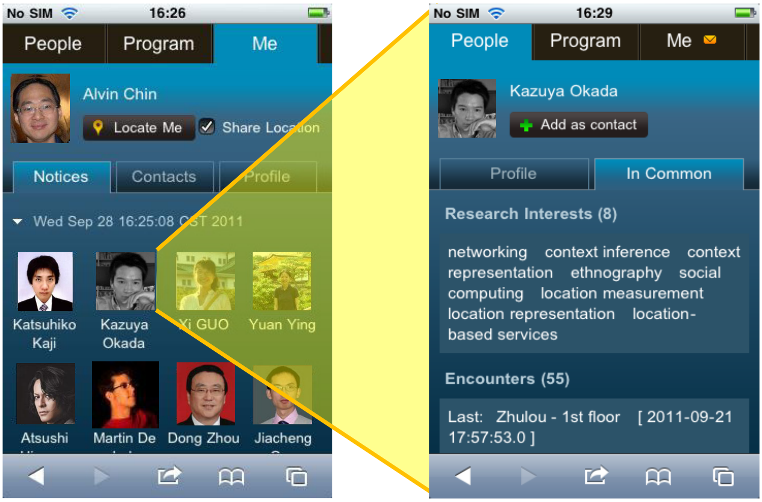
1 Introduction

An ephemeral social network indicates a social network temporarily created during an event such as a conference, game, or banquet. Such social networks are usually formed quickly and dissolve in minutes as well. Ephemeral social networks exist in both online and offline domains. In fact, these networks play an important role to expand users' social circle and strengthen social ties [16]. Different persons have very different behaviors in the ephemeral social networks. It is interesting and also important to understand what are the driving forces for persons to select targets to meet.

There has been a few related works. For example, Eagle et al. [9] studied how friend relationships are formed by tracing users' geographic information through Wi-Fi, GPS and Bluetooth. They found that friends demonstrate distinctive temporal and spatial patterns in their physical proximity and calling patterns. Crandall et al. [7] investigated how social ties between people can be inferred

* The work is supported by Nokia Research Center and is also in part supported by the Natural Science Foundation of China (No. 61073073 , No. 60973102 , No. 61170061), Chinese National Key Foundation Research (No. 60933013, No.61035004), a special fund for Fast Sharing of Science Paper in Net Era by CSTD, and National Basic Research Program of China (No. 2011CB302302).

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 613–628, 2012.



Research Interests (8)

networking context inference context representation ethnography social computing location measurement location representation location-based services

Encounters (55)

Last: Zhulou - 1st floor [2011-09-21 17:57:53.0]

Fig. 1. An ephemeral social network via the Find & Connect system at Ubicomp'11 [5]. The left figure shows the recommended users for "Chin"; the right figure shows the detailed information of a recommended user.

from co-occurrence in time and space. Tang et al. [25] developed a general learning framework for inferring the types of social ties in social networks; and [22] further extended the problem of inferring social ties across heterogeneous networks by incorporating social theories such as social balance theory and social statu theory. However, all the aforementioned work only consider the problem in normal social networks. The situation is very different in ephemeral social networks. In a normal social network, friends tend to meet together to share recent experiences. However, in an ephemeral network, people are often inclined to make new friends. For example, in an academic conference, people may want to build new research collaborations with people who they may do not know before. An interesting question is: how likely are two random persons in an ephemeral social network to gather together, and how does the likelihood depend on users' personal information and their onsite spatial information?

We use an example to clearly motivate this work. Figure 1 shows the interface of our developed Find & Connect system on a mobile phone. The system is designed for facilitating social interactions in ephemeral social networks, and has been deployed in several real scenarios including Ubicomp'11, Nokia Research Center office, and Tsinghua Centenary Celebration. Employing the Ubicomp'11 conference as the example, the system allows the user to locate friends, check attendees in surrounding areas. One important feature of the system is to recommend people to meet. The left of Figure 1 shows the recommendation results for user "Chin". The user can then see each recommended user (right figure). Obviously, an accurate recommendation algorithm should consider not only social networking information, but also the onsite location information.

We formalize the problem of inferring geographic coincidences in ephemeral social networks. The goal is to investigate the underlying patterns that drive people to meet together, and to predict how likely a geographic coincidence would happen in the near future. The problem presents a set of challenges.

- *Making new friends.* As stated before, an important objective of users joining an ephemeral social network (event) is to build new connections. It is important to predict new friendships in social networks.
- *Combining normal networks.* An ephemeral social network is not standalone. For example, attendees of an academic conference can be connected to academic social networks such as ResearchGate or Arnetminer. However, it is unclear how to combine the various normal networks for better prediction of the geographic coincidences.
- *Partially observed.* The ephemeral social network is always partially observed. Even the best organized event, there might be a portion of missing data due to various reasons, e.g., device failure and privacy protection. How to build a predictive model by considering the unlabeled data is a challenge.

To address the above challenges, we first study the behavioral patterns on how users meet together. We have found several interesting phenomena from both data distribution and social theory aspects. The duration of two persons staying at the same place and at the same time obeys a two-stage power-law distribution. Ten minutes seems to be a boundary for users to staying together. From another perspective, ephemeral social networks represent more elite-related activities: elite users tend to meet together and ordinary users are also inclined to meet elite users. We also study two important social theories, homophily and structural hole, in the ephemeral social network.

Based on the discovered behavioral patterns, we present a semi-supervised predictive framework, which incorporates the various patterns in a unified model. An efficient algorithm is developed to learn the framework. Our experiments on two different networks validate the effectiveness of the proposed methodologies. Comparing with several baseline methods using SVM and CRF, the proposed model can improve the prediction performance by 8-19% (in terms of F1-score).

2 Preliminaries

In this section, we first define the ephemeral social network and present our problem formulation. Then we describe the data sets used in our empirical study.

2.1 Problem Formulation

An ephemeral social network is a temporary and dynamic network. Generally, we can consider users from (different) normal social networks form the temporary structure and behaviors in the ephemeral social network. For example, in a game, users may form different groups based on their relationships and intimacy, while in a conference people gather in a technical session according to their interest.

Let $G = (V, E, \mathbf{W})$ represent a normal social network, where V is a set of users, $E \subset V \times V$ is a set of relationships between users, and $\mathbf{W}$ is an attribute matrix associated with users V. An ephemeral social network can be defined as $G'(t) = (U^t, \mathbf{X}^t, Y^t)$, where $U \subset V$ is a subset of V indicating users forming the ephemeral social network come from a normal social network, $\mathbf{X}^t$ denotes an ephemeral attribute matrix for users in U^t, and Y^t denotes a set of user behaviors we want to predict, e.g., whether a user will join a seminar.

Without loss of generality, we employ the ephemeral network built in the UbiComp 2011 conference as the example to define our problem. Users of the ephemeral network are researchers from universities and companies. Their corresponding normal network can be defined as the coauthor network. The ephemeral attributes include where the user is, when the user will give a talk, what the user is doing, etc.

A usual predictive task in an ephemeral network is to predict users' future behavior by leveraging the normal social network and users' ephemeral attributes. In this work, we consider the problem of geographic coincidence prediction. The objective is to predict whether two users will meet together in the near future. Formally, the problem can be defined as:

Problem 1. **Geographic coincidence prediction.** Given a normal network $G = (V, E, \mathbf{W})$ and an ephemeral network $G'(t) = (U^t, \mathbf{X}^t, Y^t)$, the goal is to learn a predictive function:

$$f : \{G'(t), G\} \rightarrow Y^{(t+1)}$$

where $y_{ij}^{(t+1)} \in \{0, 1\}$ indicates whether user u_i and u_j will meet at time $(t + 1)$.

Roughly speaking, we try to infer whether two users will gather at approximately the same place and at approximately the same time. More accurately, we say that two users u_i and u_j have a geographic coincidence (i.e., $y_{ij} = 1$) if their distance is shorter than a constant (D meters) for more than M minutes. The definition of geographic coincidence might be different in some other scenario. For example, in the MIT's Reality data set, users' coincidence are measured by Bluetooth devices.

2.2 Data Sets

We study the problem of geographic coincidence prediction on two different types of social networks: UbiComp and Reality.

UbiComp. The UbiComp data set is collected by Find & Connect[1], a social network platform built for participants of conferences or meetings for finding conference resources and people and connecting with them. With a positioning system based on RFID or Wi-Fi, Find & Connect records the indoor location data for each user and provides indoor location-based services such as finding where the paper or session is being held, who are the people attending the sessions, where people are in the conference and when, and where was the last time that two users have met. Thereby we are able to acquire logs of physical proximity, which implies a probable

[1] An ephemeral social networking system developed in Nokia Research Center.

encounter and interaction between users, as well as social networking connections. The system has been deployed at the UIC 2010 conference [28], Nokia GCJK internal marketing event and UbiComp 2011[5].

We use the UbiComp data set, which consists of 234 users and 69,844 location logs during the 3-day conference. The data set is divided into time intervals by day. The proximity encounters are recorded from mining locations of users equipped with RFID tags using RFID readers and a modified version of the LANDMARC algorithm [19]. Given this, we say that two users u_i and u_j have a geographic coincidence if their distance becomes shorter than D meters at a specific time, and remains within the range of $[0, D)$ for more than M minutes.[2]

Since most attendees of UbiComp are academic researchers, we can acquire their publication lists and coauthor relationships by their names in ArnetMiner[3] [24], which consists of 1,756,147 authors and 1,813,514 publications as well as the coauthor relationships between users. Finally, out of 234 UbiComp users, 206 of them are found in ArnetMiner. We thereby obtain their research profiles including their publications, co-authorship and attended conferences.

Reality. The Reality data set is collected from 106 users from September 2004 to June 2005 in MIT. A pre-installed software on each user's mobile phone will record their communication logs as well as physical proximity logs. The communication logs include voice calls and short messages. The physical proximity logs are recorded by the Bluetooth sensor, which scans for other contacts on average every 5 minutes. If the Bluetooth sensor of a user detects another sensor at a certain time, a physical proximity event between these two users will be recorded. Reality data set contains 162,700 communication logs and more than 4 millions physical proximity logs in total. Similarly, the Reality data set is divided into time intervals by day.

In addition to the geographic coincidences, the Reality data also contains the friendships between two users collected by querying the users, which form a friendship network between all the users. In the Reality data set, we directly regard each proximity log as a geographic coincidence since the detection range of a Bluetooth sensor is approximately 5-10 meters, which is close enough for a geographic coincidence.

3 Observations

In this section, we conduct the following observations based on the UbiComp data in order to get a better understanding on the users' behavioral patterns and structural properties of ephemeral social network:

- *Two-stage power-law distribution.* We analyze the duration distribution of geographic coincidences and find that it satisfies a certain two-stage power-law distribution.

[2] We empirically set $D = 3$ and $M = 10$, which is based on the observation in [11] and the "ten-minutes" phenomenon we discovered in observations (Cf. §3).

[3] http://arnetminer.org

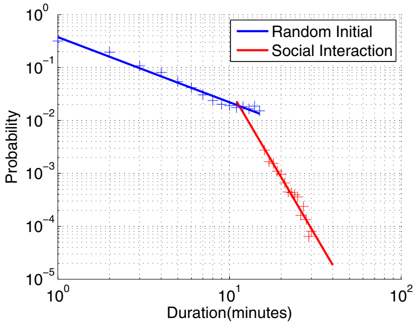


Fig. 2. Duration distribution

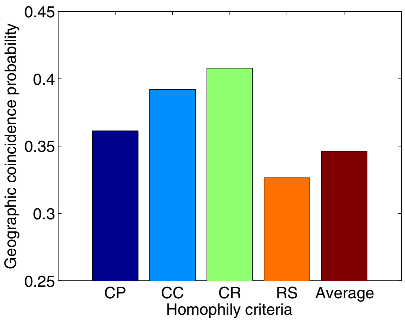


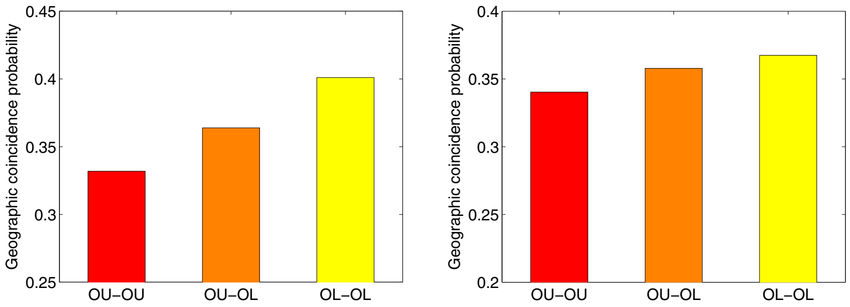
Fig. 3. Link Homophily

- *Link homophily.* How does the user similarity influence the geographic coincidences pattern?
- *Opinion leaders.* What is the role played by opinion leaders in ephemeral social network?
- *Structural hole.* Do users who span a structural hole have geographic coincidences with different people?

Two-Stage Power-Law Distribution. We first study the duration patterns of two users staying at approximately the same place and at approximately the same time. Figure 2 plots the distribution in a log-log space. It can be interestingly seen that the distribution can be described using a two-stage power-law and 10 minutes seems to be an inflexion point. When the duration time is less than 10 minutes, the exponent of the corresponding power-law is -1.2315, while, when the duration time increases to more than 10 minutes, the exponent becomes -5.5221. The phenomenon implies that a large portion of coincidences might be random based on users' location. For example, acquaintances generally say hello when they meet and make small talk (less than 10 minutes). On the other hand, targeted meetings may last a longer time.

Based on this observation, we set the duration threshold as $M = 10$ for the definition of geographic coincidence on the UbiComp data set. That is to say, on the UbiComp data set we only consider geographic coincidences longer than 10 minutes since they are more likely to indicate actual social interactions.

Link Homophily. The principle of homophily [16] points out that users with higher similarity are more likely to establish relationships. In this work we mainly study the similarity in research area since most of the users are researchers and they attend the conference in order to get feedback or establish new collaborations in academia. The following criteria are employed to measure users' research similarity: (1) *Coauthored paper count (CP)*: It counts the coauthored publication number for each pair of users; (2) *Common coauthor count (CC)*: It counts the number of common coauthors between two users; (3) *Common conference ratio (CR)*: We construct conference vectors for all users with their attendance times of different conferences. The common conference ratio is the cosine similarity of



(a) Publication count

(b) H-index

Fig. 4. Observation of geographic coincidences between opinion leaders (OL) and ordinary users (OU)

two users' conference vectors; (4) *Research similarity (RS)*: Jaccard similarity of the research interests of two users.

We rank all the user pairs by the above criteria and calculate the geographic coincidence probability of the top 600[4] pairs of users. The average geographic coincidence probability is also calculated for comparison, as shown in Fig. 3. We can observe that user pairs with highest CP, CC, or CR are more likely to have geographic coincidences than average. These results are expected. Users with more coauthored papers have direct connections between them and thus are more likely to meet each other; more common coauthors implies a strong effect of triadic closure [10], which influences the geographic coincidence probability and attending more common conferences increases their chance to know each other. However, a surprising observation is that geographic coincidence probability of user pairs with highest research similarity are approximately 2% lower than the average probability. This result indicates that attendees of an academic conference may tend to talk with people that have different research interests in order to get new ideas.

Opinion Leader. The two-step flow theory [2,14,16] suggests that ideas usually flow first to "opinion leaders" and then to more people from them. There are several algorithms to detect opinion leaders in social networks. In this work we use two different indicators to define opinion leaders: publication count and H-index. We rank all the users by their publication count or H-index and take the top 25% as opinion leaders. Fig. 4 presents the comparison of geographic coincidence probability between different types of user pairs. It is clearly shown that ordinary users (OU) and opinion leaders (OL) are more likely to have a geographic coincidence than two ordinary users, which implies that people tend to communicate with opinion leaders. We also find that two opinion leaders have the highest probability of geographic coincidence. This is expected because in an academic conference, opinion leaders are more willing to exchange ideas and hence have more direct interactions.

[4] Probability of top 200, 400 pairs of users yields similar results.

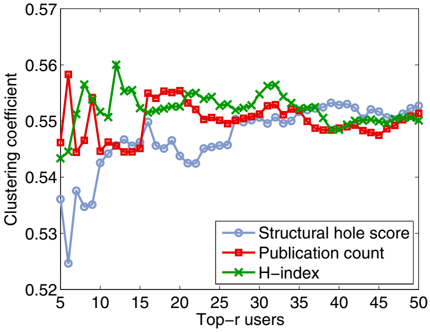


Fig. 5. Clustering coefficient

Structural Hole. In a social network, a person is called to span a *structural hole* if she is connected to two people in different parts of networks that are otherwise not well connected to each other [3]. It is claimed that such nodes have an informational advantage with connection to people who are not linked to each other, and hence are exposed to a more diverse source of ideas. An interesting question is, whether a person who spans a structural hole in coauthor network will also present a higher diversity in its geographic coincidence pattern? In this paper, we simply define node A's "structural hole score" in a coauthor network by the number of author pairs (B, C) which satisfies that A is the only common coauthor. We rank all the users by their structural hole score, and calculate the average clustering coefficient of top-r users over ephemeral social networks of all the time intervals in UbiComp data set. We also rank the users by publication count and H-index to provide a comparison to opinion leaders. The result is presented in Fig. 5. It is shown that users with structural hole score ranking in the top 20 tend to have lower clustering coefficient (confirmed by paired t-test with 95% significance), but it turns out to be close to the average when taking the top 50 users into account. The clustering coefficient of opinion leaders, however, always remain consistent with average level. It indicates that users who have a higher structural hole score also tend to have geographic coincidences with a wide variety of people, but the difference is slight since in an ephemeral social network, a larger proportion of users seeks for new relationships and hence have geographic coincidences with various people.

4 Factor Graph Model

We employ a factor graph model to predict the geographic coincidences between users. The basic idea is to construct a graphical model by modeling each pair of users as a node. We then define different types of factor functions to incorporate different factors into the prediction task, and define an objective function based on the joint probability of the factor functions. The model can be trained by optimizing the objective function.

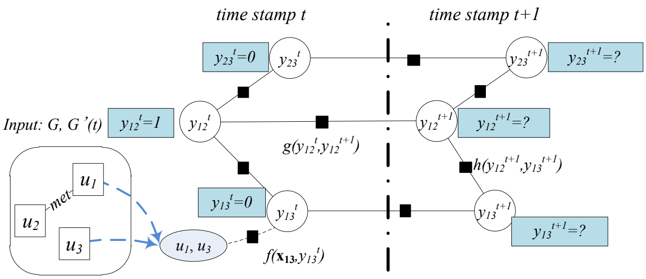


Fig. 6. A graphical representation of factor graph model

As Fig. 6 demonstrates, at time t, we map the event of geographic coincidence of every pair of users (u_i, u_j) as a node y_{ij}^t in our graphical model, corresponding to an event to predict in the ephemeral social network $G'(t)$. We use Y to represent the global set of all y_{ij}^t. For labeled samples, when users have geographic coincidence we have $y_{ij}^t = 1$, otherwise $y_{ij}^t = 0$; for unlabeled samples, we leave $y_{ij}^t =?$ to predict. The factor graph model was previously used for inferring social ties in social networks [25].

We define three different kinds of factor functions as follows:

- **Attribute factor function** $f(\mathbf{x}_{ij}, y_{ij}^t)$. It incorporates the attribute value $\mathbf{x}_{ij}$ of each pair of users corresponding to y_{ij}^t, where $\mathbf{x}_{ij} = [\mathbf{w}_{i\cdot}, \mathbf{w}_{j\cdot}]$ combines the attribute vector of both users.
- **Temporal correlation factor function** $g(y_{ij}^t, y_{ij}^{t+1})$. It represents the temporal dependencies between the geographic coincidences indicated by y_{ij}^t and y_{ij}^{t+1}.
- **Social correlation factor function** $h(Y_c^t)$. Y_c^t represents a clique which consists of a set of y_{ij}^t. It leverages the social correlation between user pairs.

The three factor functions can be instantiated in different ways. In this work, we define them as exponential-linear functions. Formally, we define the attribute factor function as

$$f(\mathbf{x}_{ij}, y_{ij}^t) = \frac{1}{Z_1} \exp\{\alpha^T \mathbf{\Phi}(\mathbf{x}_{ij}, y_{ij}^t)\} \tag{1}$$

where α is the weighting vector; $\mathbf{\Phi}(\mathbf{x}_{ij}, y_{ij}^t)$ is the feature vector function.

The temporal correlation factor function can be defined as

$$g(y_{ij}^t, y_{ij}^{t+1}) = \frac{1}{Z_2} \exp\{\beta^T \mathbf{g}(y_{ij}^t, y_{ij}^{t+1})\} \tag{2}$$

where β is the weighting vector; $\mathbf{g}(y_{ij}^t, y_{ij}^{t+1})$ is an indicator function.

We define the social correlation factor function in a similar way

$$h(Y_c^t) = \frac{1}{Z_3} \exp\{\lambda^T \mathbf{h}(Y_c^t)\} \tag{3}$$

where λ is the weighting vector; $\mathbf{h}(Y_c^t)$ is an indicator function, taking the geographic coincidences of a clique of user pairs as input. Z_1, Z_2 and Z_3 are normalizing factors. This definition of factor function has been used in a graphical models such as Markov Random Fields [12] or Conditional Random Fields [15].

The joint distribution over all the Y can be written as

$$P(Y|G,G') = \frac{1}{Z}\exp\{\sum_t \sum_{y_{ij}^t} \alpha^T \mathbf{\Phi}(\mathbf{x}_{ij}, y_{ij}^t) + \sum_{i,j}\sum_t \beta^T \mathbf{g}(y_{ij}^t, y_{ij}^{t+1}) + \sum_t \sum_{Y_c^t} \lambda^T \mathbf{h}(Y_c^t)\} = \frac{1}{Z}\exp\{\theta^T \mathbf{S}\} \quad (4)$$

where $\theta = [\alpha^T, \beta^T, \lambda^T]^T$ is the parameter vector;
$\mathbf{S} = [\sum_t \sum_{y_{ij}^t} \mathbf{\Phi}(\mathbf{x}_{ij}, y_{ij}^t), \sum_{i,j}\sum_t \mathbf{g}(y_{ij}^t, y_{ij}^{t+1}), \sum_t \sum_{Y_c^t} \mathbf{h}(Y_c^t)]^T$ denotes all the features and Z is the normalizing factor.

Model Learning. We learn the FGM by estimating the parameter configuration θ to optimize the log-likelihood of observed data. The observed data could be incomplete and thus pose challenges to model learning. We regard the entire factor graph as a partially labeled graph. Let Y_L denote the set of known geographic coincidences, and Y_U as the set of unknown geographic coincidences. The learning task can be formally described as to find a parameter configuration θ^* such that $\theta^* = \text{argmax}_\theta P(Y_L|G,G')$.

We define the log-likelihood as the objective function

$$\begin{aligned}\mathcal{O}(\theta) &= \log P(Y_L|G,G') = \log \sum_{Y|Y_L} \exp \theta^T \mathbf{S} - \log Z \\ &= \log \sum_{Y|Y_L} \exp \theta^T \mathbf{S} - \log \sum_Y \exp \theta^T \mathbf{S} \end{aligned} \quad (5)$$

A gradient decent method (Newton-Raphson method) is used to optimize Eq. 5. The gradient for each parameter is

$$\begin{aligned}\frac{\partial \mathcal{O}(\theta)}{\partial \theta} &= \frac{\sum_{Y|Y_L} \exp \theta^T \mathbf{S} \cdot \mathbf{S}}{\sum_{Y|Y_L} \exp \theta^T \mathbf{S}} - \frac{\sum_Y \exp \theta^T \mathbf{S} \cdot \mathbf{S}}{\sum_Y \exp \theta^T \mathbf{S}} \\ &= \mathbb{E}_{P(Y|Y_L,G,G')}\mathbf{S} - \mathbb{E}_{P(Y|G,G')}\mathbf{S}\end{aligned} \quad (6)$$

We use Loopy Belief Propagation (LBP) to approximate the gradient and update θ iteratively.

Predicting Geographic Coincidences. With the learned parameter configuration θ, the prediction task is to find a Y_U^* which optimizes the objective function, i.e., $Y_U^* = \text{argmax}_{Y_U} P(Y|G,G')$.

We employ similar methodology in this optimization task. Instead of calculating the joint probability, we calculate the marginal probability for each $y_{ij}^{(t+1)}$ and predict them as positive when the marginal probability is greater than 0.5, otherwise the event will be predicted as negative.

Table 1. Statistics of UbiComp and Reality data sets

Data set	Users	Labeled samples	Unlabeled samples
UbiComp	243	17,391	23,871
Reality	106	7,384	2,140

5 Experimental Results

5.1 Experimental Setup

Data Sets. We validate the effect of our proposed model on two different data sets: *UbiComp* and *Reality*, and compare the result with two baseline methods. A brief statistics of the data sets is shown in Table 1.

UbiComp data set includes user location logs on September 19th and September 21st. In this work, we divide the data set into two time intervals, namely the two days of the conference. We regard all the geographic coincidences on September 19th as labeled and predict the geographic coincidences on September 21st.

For Reality data, we select 12 consecutive days, each with more than 100 communication logs for our experiments. Then we define the first 10 days as labeled. The task is to predict the geographic coincidences in the last 2 days.

Baseline Methods. We define two baseline methods for the geographic coincidences task.

- *SVM.* This method only uses the users' attribute to train SVM and to predict the geographic coincidences.
- *CRF.* We consider the time correlation and establish sequential conditional random fields for each user pair.

We evaluate the performance of geographic coincidence inference in terms of precision, recall and F1-score.

Factor Definitions. For both data sets, we define the temporal correlation factors between two consecutive time intervals for each user pair.

In UbiComp data set, we also define four different types of social correlation factors according to the principle of homophily (Cf. Section 3): if two users are similar in some aspects, they will be more likely to have geographic coincidence with the same person. To define the social correlation factors based on homophily of coauthored paper count (CP), we first rank all the user pairs by CP and select those within the top 150, denoted by (u_i, u_j). Then for every other user u_n, we add social correlation factors *CPInf* between e^k_{in} and e^k_{jn}. The other three homophily-based social correlation factors *CCInf*, *CRInf* and *RSInf* can be defined similarly.

In Reality data set, we define social correlation factors based on the structural balance theory [10]. It suggests that people in a social network tend to form into a balanced network structure. To be specific, for a triad, the balance theory

Table 2. Prediction performance comparison(%)

Date set	Method	Precision	Recall	F1-score
UbiComp	SVM	**34.5**	20.4	25.6
	CRF	33.2	39.4	36.0
	FGM	34.0	**65.4**	**44.7**
Reality	SVM	84.1	64.4	72.9
	CRF	73.6	**85.8**	79.2
	FGM	**85.1**	81.0	**83.0**

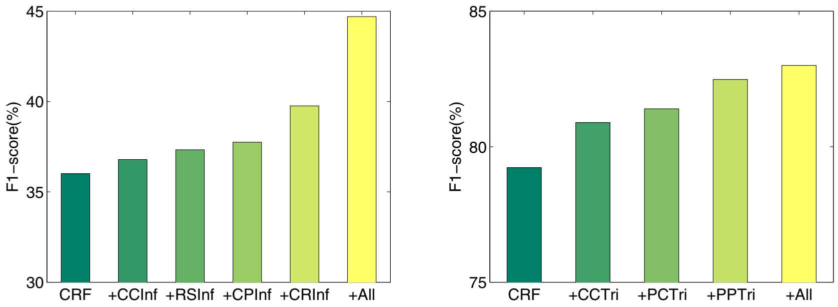
claims that all of the three users or only one pair of them should be friends. We employ two kinds of connection, physical proximity connection and calling connection (voice call or SMS), to identify triads. Then for each triad u_i, u_j, and u_k, we establish social correlation factors between every two user pairs. Since there are two types of connections, we can define three different social correlation factors regarding the connection types of the involved two user pairs: *CCTri* (both calls), *PCTri* (one call, one physical proximity) and *PPTri* (both physical proximity).

5.2 Results and Discussion

Performance Comparison. We compare the prediction performance between our methods and the baselines, as shown in Table 2. It is shown that our model outperforms other methods in both two data sets. In UbiComp data set, FGM achieves an improvement of approximately 8-19% in terms of F1-score compared to the baselines, and also improves recall by approximately 26-45%. Although SVM shows a higher precision of 34.5%, the precision of the proposed model is very close to the baseline (34.0%). In Reality data set, FGM also gives a rise of 4-10% compared to the baselines in terms of F1-score. In addition, FGM achieves the highest precision among all the methods. We can also observe the effect of time correlation, employed by CRF. The time correlation factor improves the F1-score of CRF by about 10% in UbiComp data set and approximately 7% in Reality data set.

Contribution of Social Correlation Factors. To further investigate the contribution of different social correlation factors in the prediction task, we remove all the social correlation factors and evaluate the performance by adding each of them individually into the model. Thereby we can measure their contribution by the improvement they achieve to F1-score, as shown in Fig. 7.

It is shown that in both data sets, all social correlation factors improve the performance. In UbiComp data set, CRInf factor contributes the most to F1-score amongst the four social correlation factors by an average improvement of 3%. It implies that users who often attend common conferences may have a stronger implicit correlation since they probably have been in the same ephemeral social network before. Its effect is even stronger than those with explicit coauthorship (CPInf). The effect of CCInf, RSInf and CPInf factors are also observable.



(a) UbiComp (b) Reality

Fig. 7. Analysis of social correlation factors on F1-score

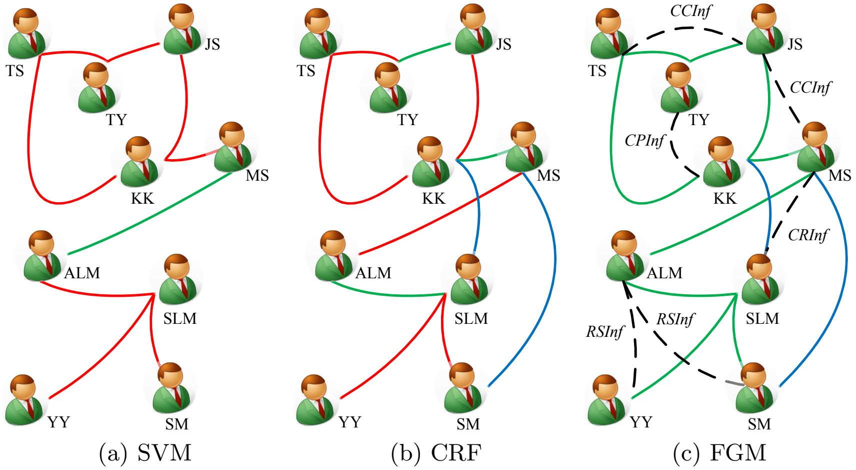
In Reality data set, it can be observed that PPTri achieves the highest improvement of approximately 3%. It implies that ephemeral social network probably obeys the structural balance theory and indeed helps improve the performance. PCTri and CCTri also contribute significantly to the improvement of performance. It indicates that joint with relationships in normal social network such as mobile social network, triadic social correlation factors based on structural balance theory still contributes to the prediction performance.

Case Study. We further conduct a case study to investigate why our proposed model outperforms other baseline methods. Fig. 8 presents the prediction result on a subset of UbiComp data generated by three different approaches: SVM, CRF and FGM. Green solid lines represent true positive samples; red solid lines for false negative samples and blue solid lines for false positive samples. In addition, we use black dash lines to point out the social correlations between users.

It can be observed that CRF tends to predict more geographic coincidences than SVM with the help of time correlation. It successfully detects more geographic coincidences (e.g. JS-TY and MS-KK), albeit few of them are incorrect (Cf. Fig. 8(b)). Our proposed approach further leverages the social correlation factors to improve the prediction result. For example, when MS and JS have a higher common coauthor count, geographic coincidences of MS and KK may increase the chance of a geographic coincidence between JS and KK. Our proposed model is able to capture such social correlations and infer the geographic coincidences between KK and JS from the prediction between MS and KK. The social correlation factors benefit the prediction result of FGM by significantly improving the recall, as shown in Fig. 8(c).

6 Related Work

Dynamic Behavior Analysis. There are several works on social dynamic behavior analysis. Zhang et al. [26] proposed a dynamic continuous factor graph model to predict users' emotion states. Tan et al. [21] proposed a noise tolerant model for predicting user's actions in online social networks. Tang et al. [23]



(a) SVM (b) CRF (c) FGM

Fig. 8. Case study

proposed a topical affinity propagation to quantify the social influence between users. However, these works did not leverage location information, while we focus on predicting geographic coincidences.

User Mobility Analysis. Quite a few works on user mobility analysis have been conducted. Li et al. [17] designed a hierarchical-graph-based similarity measurement for estimating user similarity based on their location history. Liu et al. [18] proposed an approach to utilize information of mobile objects for the clustering task. Qian et al. [20] explore co-location mining pattern with dynamic neighborhood constraint. However, rather than analysis of user mobility, we focus on a prediction problem. Cho et al. [6] develop a Gaussian model by incorporating periodicity and influence of social network structure to predict human location tracks. Crandall et al. [7] studies geographic coincidences between users to infer social ties, while our work focus on prediction of geographic coincidences from social network. Zheng et al. [27] used a graph-based algorithm to infer user mobility based on GPS data. Tang et al. [25] developed a general learning framework for inferring the types of social ties in social networks; and [22] further extended the problem across heterogeneous networks. But none of these works provide an approach for prediction of interpersonal geographic coincidences.

Physical Proximity Analysis. Physical proximity has been employed in many works to quantify users' behaviors. Eagle et al. [8] use GPS on mobile phones to analyze proximity of the users in order to present the properties of users' location tracks. However, different from tracking users' mobility, we aim to predict geographic coincidences between users in this work. There is also a host of conference proximity analysis in current literature. Isella et al. [13] use RFID badges to collect face-to-face proximity data of individuals at a scientific conference, and analyze its static and dynamic properties. Atzmueller et al. [1] explore different roles of participants in a conference by examining their face-to-face interaction patterns.

Similarly, Cattuto et al. [4] collect data from the office environment and academic congress.

7 Conclusion

In this paper, we formally define the ephemeral social network and study to which extent we can predict geographic coincidences in an ephemeral social network. We conduct a series of observations on an ephemeral social network extracted from a data collected during an academic conference (UbiComp 2011). Based on link homophily, opinion leader and structural hole, we show the interplay between the normal social network (coauthor network) and users' behavioral pattern in the ephemeral social network. We then propose a Factor Graph Model (FGM) for the prediction task. Experimental results show that our model outperforms the baseline on two data sets: UbiComp and Reality. Further analysis also suggests that social correlation factors help improve the performance.

A limitation of this work is that a geographic coincidence does not necessarily indicate an actual social interaction, e.g. conversation or discussion. We carefully select the parameters so that extracted geographic coincidence are very likely to be accompanied with actual social interaction, but the real situation is hard to detect without collecting additional context. Another flaw is the requirement of labeled data since we use supervised learning for our model. An unsupervised learning approach would further reduce the cost of geographic coincidences prediction.

References

1. Atzmueller, M., Doerfel, S., Hotho, A., Mitzlaff, F., Stumme, G.: Face-to-face contacts during a conference: Communities, roles, and key players. In: Workshop MUSE at ECML/PKDD (2011)
2. Booth, A.: Personal influence networks and participation in professional association activities. The Public Opinion Quarterly 33(4), 611–614 (1969)
3. Burt, R.S.: Structural holes: The social structure of competition. Harvard University Press, Cambridge (1992)
4. Cattuto, C., Van den Broeck, W., Barrat, A., Colizza, V., Pinton, J.-F., Vespignani, A.: Dynamics of person-to-person interactions from distributed rfid sensor networks. PLoS ONE 5(7), e11596 (2010)
5. Chin, A., Xu, B., Yin, F., Wang, X., Wang, W., Fan, X., Hong, D., Wang, Y.: Using proximity and homophily to connect conference attendees in a mobile social network. Accepted to the 2nd International Workshop on Sensing, Networking and Computing with Smartphones, pp. 1–9. IEEE Computer Society (2012)
6. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: KDD, pp. 1082–1090 (2011)
7. Crandall, D., Backstrom, L., Cosley, D., Suri, S., Huttenlocher, D., Kleinberg, J.: Inferring social ties from geographic coincidences. PNAS 107(52), 22436 (2010)
8. Eagle, N., Pentland, A.: Social serendipity: Mobilizing social software. IEEE Pervasive Computing 4(2), 28–34 (2005)
9. Eagle, N., Pentland, A., Lazer, D.: Inferring friendship network structure by using mobile phone data. PNAS 106(36), 15274 (2009)

10. Easley, D.A., Kleinberg, J.M.: Networks, Crowds, and Markets - Reasoning About a Highly Connected World. Cambridge University Press (2010)
11. Hall, E.T.: A system for the notation of proxemic behaviour. American Anthropologist 65, 1003–1026 (1963)
12. Hammersley, J.M., Clifford, P.: Markov field on finite graphs and lattices (1971) (unpublished manuscript)
13. Isella, L., Barrat, J.S.A., Cattuto, C., Pinton, J.-F., den Broeck, W.V.: What's in a crowd? analysis of face-to-face behavioural networks. Journal of Theoretical Biology, 166–180 (2010)
14. Katz, E.: The two-step flow of communication: An up-to-date report on an hypothesis. The Public Opinion Quarterly 21(1), 61–78 (1957)
15. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: ICML, pp. 282–289 (2001)
16. Lazarsfeld, P.F., Berelson, B., Gaudet, H.: The People's Choice. How the Voter Makes up his Mind in Presidential Campaign. Columbia University Press, New York (1944)
17. Li, Q., Zheng, Y., Xie, X., Chen, Y., Liu, W., Ma, W.: Mining user similarity based on location history. In: Workshop on Advances in Geographic Information Systems (2008)
18. Liu, S., Liu, Y., Ni, L.M., Fan, J., Li, M.: Towards mobility-based clustering. In: KDD, pp. 919–928 (2010)
19. Ni, L.M., Liu, Y., Lau, Y.C., Patil, A.P.: Landmarc: indoor location sensing using active rfid. Wireless Networks 10, 701–710 (2004)
20. Qian, F., He, Q., He, J.: Mining Spatial Co-location Patterns with Dynamic Neighborhood Constraint. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 238–253. Springer, Heidelberg (2009)
21. Tan, C., Tang, J., Sun, J., Lin, Q., Wang, F.: Social action tracking via noise tolerant time-varying factor graphs. In: KDD, pp. 1049–1058 (2010)
22. Tang, J., Lou, T., Kleinberg, J.: Inferring social ties across heterogeneous networks. In: WSDM 2012, pp. 743–752 (2012)
23. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: KDD, pp. 807–816 (2009)
24. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: Extraction and mining of academic social networks. In: KDD, pp. 990–998 (2008)
25. Tang, W., Zhuang, H., Tang, J.: Learning to Infer Social Ties in Large Networks. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 381–397. Springer, Heidelberg (2011)
26. Zhang, Y., Tang, J., Sun, J., Chen, Y., Rao, J.: Moodcast: Emotion prediction via dynamic continuous factor graph model. In: ICDM, pp. 1193–1198 (2010)
27. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.: Understanding mobility based on GPS data. In: Ubiquitous Computing/Handheld and Ubiquitous Computing, pp. 312–321 (2008)
28. Zhu, L., Chin, A., Zhang, K., Xu, W., Wang, H., Zhang, L.: Managing Workplace Resources in Office Environments through Ephemeral Social Networks. In: Yu, Z., Liscano, R., Chen, G., Zhang, D., Zhou, X. (eds.) UIC 2010. LNCS, vol. 6406, pp. 665–679. Springer, Heidelberg (2010)

Pedestrian Quantity Estimation with Trajectory Patterns

Thomas Liebig, Zhao Xu, Michael May, and Stefan Wrobel

Fraunhofer IAIS
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
{Thomas.Liebig,Zhao.Xu,Michael.May,Stefan.Wrobel}@iais.fraunhofer.de

Abstract. In street-based mobility mining, traffic volume estimation receives increasing attention as it provides important applications such as emergency support systems, quality-of-service evaluation and billboard placement. In many real world scenarios, empirical measurements are usually sparse due to some constraints. On the other hand, pedestrians generally show some movement preferences, especially in closed environments, e.g., train stations. We propose a Gaussian process regression based method for traffic volume estimation, which incorporates topological information and prior knowledge on preferred trajectories with a trajectory pattern kernel. Our approach also enables effectively finding most informative sensor placements. We evaluate our method with synthetic German train station pedestrian data and real-world episodic movement data from the zoo of Duisburg. The empirical analysis demonstrates that incorporating trajectory patterns can largely improve the traffic prediction accuracy, especially when traffic networks are sparsely monitored.

Keywords: Pedestrian Quantity Estimation, Trajectory, Gaussian Process Regression, Graph Kernels.

1 Introduction

Estimation of traffic volumes is a common task for street based traffic and the achieved values are highly interesting for risk analysis, quality of service evaluation, location ranking and mobility analysis applications. Particularly, for pedestrians traffic, knowledge on people's presence offers a vast chance for improvement of the signage and the infrastructure. Facilities provided to people depend on pedestrian movements and volumes. To give a few general examples: locations of information desks, shops or toilettes depend on the quantity of persons; path-widths of the corridors in a stadium depend on people's quantity as well, mobile phone networks are planned according to the expected movements and even locations of advertisement billboards are placed such that they are potentially noticed by as many pedestrians as possible. Modelling the pedestrian quantities gives indispensable insights on visitor preferences and motivations at a particular public event or site and thus supports creation of intelligent environments.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 629–643, 2012.

In this work we focus on the estimation of traffic volumes for pedestrians within closed environments. These are sites or buildings which have in common, that no people reside inside but all present people leave after some time period. Thus, these closed environments have dedicated entrances and exits. Prominent examples are train stations, terminals, shops, shopping malls, parks as well as zoos. As shown in the previous examples, knowledge of pedestrian movement provides indispensable benefits to safety, marketing as well as infrastructural applications. Thus, over the past years many sensor technologies to fetch empirical measurements and record pedestrian volumes have been developed (most popular ones are video surveillance, laser beams and Bluetooth sensors). However, empirical measurements are usually rare due to some constraints, e.g., budget limitations. This arises the following questions.

– How can values on pedestrian quantities be estimated from few empirical measurements?
– At which places should a constrained number of quantity sensors be located?

Often, available data is limited to few measurements and some prior knowledge, e.g., floor plan sketches, knowledge on preferred routes by local domain experts. Incorporating prior knowledge is thus essential to address the above challenges. However there are few approaches taking into account the trajectory patterns, although pedestrians generally show some move preferences [1,2], especially in closed environments, e.g., train stations. Consider for example an average daily traffic (ADT) prediction problem with traffic networks consisting of only one junction. As shown in Figure 1, a T-junction occurs in a wide corridor that goes straight. At the junction a small corridor intersects and an expert knows that it is most likely for persons to continue their walk straight ahead in the main corridor. Assume further to have a frequency sensor placed in the main corridor which measures a known amount of people within considered time interval. Under these circumstances, existing traffic volume estimation methods, e.g., k-nearest neighbour and standard Gaussian process regression, do not take into account the expert knowledge and thus may not effectively provide accurate estimations for the side corridor.

We propose a traffic volume estimation method based on Gaussian process regression, which incorporates topological information and the expert knowledge on preferred trajectories with a trajectory pattern kernel. By exploring trajectory patterns, our method can also effectively elicit most informative sensor placements. We demonstrate the advantages of our approach with two applications. The first one is pedestrian mobility analysis in German train stations. As the data available is some statistical analysis on the network characteristics. We draw synthetic (but realistic, see Section 4) traffic networks and pedestrian movement based on these analysis. Secondly, our approach is applied to the real world scenario at the zoo of Duisburg (Germany). The pedestrian mobility data of the visitors was collected with Bluetooth tracking technology [3]. The empirical analysis demonstrates that incorporating trajectory patterns can largely improve the traffic prediction accuracy, especially when traffic networks are sparsely monitored. Our work contributes an extensive approach to the pedestrian volume

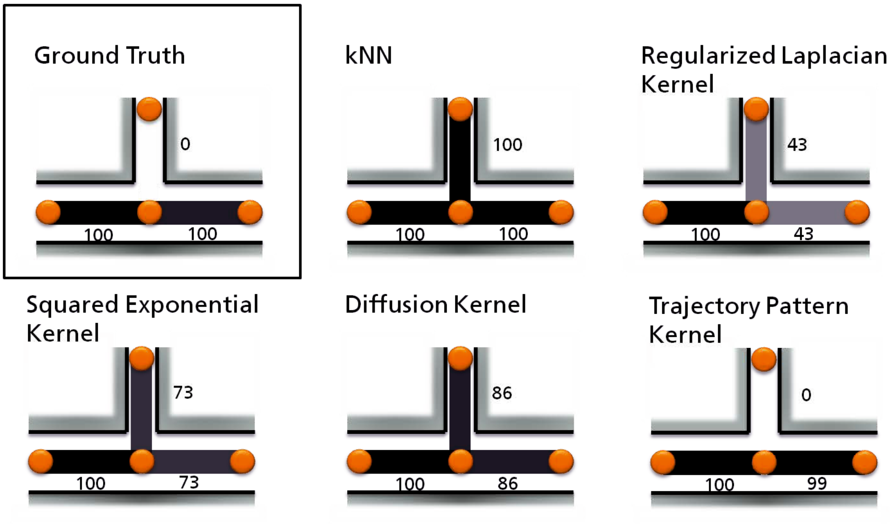

Ground Truth
0
100
100
kNN
100
100
100
Regularized Laplacian Kernel
43
100
43
Squared Exponential Kernel
73
100
73
Diffusion Kernel
86
100
86
Trajectory Pattern Kernel
0
100
99

Fig. 1. T-junction example. Main corridor is horizontal. Expert knowledge which presumes that people walk straight ahead in the main corridor is given. Left corridor frequency measurement is given. Numbers denote relative frequencies in percent.

estimation problem and it provides an efficient, applicable solution to industrial real world scenarios.

The rest of the paper is organized as follows. Section 2 gives an overview of related work. Then we describe the proposed approach in Section 3. Empirical analysis and real world applications are presented in Section 4. Finally, we give our conclusions and discussions on future work in Section 5.

2 Related Work

Existing literature distinguishes between average daily traffic (ADT) estimation and average annual daily traffic (AADT) estimation. Whereas AADT focuses on estimation of a traffic volume depending on the day of the year, ADT estimation provides an average for a particular day. Naïve approach for AADT estimation is utilization of ordinary linear regression (OLR) [4]. Street segment attributes (e.g. number of lanes or function classes) are multiplied by weights which are subject for least squares regression. Improvements of this technique were achieved by respecting the geographical space by usage of geographical weighted regression (GWR) [4] and by application of k-nearest neighbor approaches (kNN) [5]. In [6] the AADT prediction of kNN for a particular location is improved by weighting measurements by their temporal distance to the prediction time. This approach showed better results than application of Gaussian maximum likelihood (GML) approaches for weighting of the historical data points. Recent improvements to kNN non parametric regression were made in [7]. Although performing the k-nearest neighbor search in the attribute space of the street segments, this approach selects the spatially closest neighbours, as they have highest impact. In [7–9] the ADT estimation problem is addressed as business critical industrial data mining use case as the pricing in the outdoor advertisement sector in Germany and Switzerland relies on the estimated values [7]. Their proposed algorithm is a spatial k-nearest neighbour (S-kNN) approach that incorporates geometric distances for estimation of an unknown segment. The closer a measured segment is to an unmeasured one, the higher its impact. This is similar to the Kriging approach described in [10] but goes beyond it, as just the k-nearest neighbours were used for prediction.

The regression approaches in [11–13] are motivated by outdoor advertisement use cases. In contrast to the previously described methods their approaches operate in the space of the possible routes instead of the segment-attribute space. After an extensive path enumeration step, this work checks every possible path on plausibility and considers the resulting set of plausible paths for path frequency estimation using a least squares regression at the measurement locations. That approach contains a basic assumption on pedestrian route choice, namely pedestrians prefer the shortest path to travel from one location to another. But in some scenarios this assumption does not hold [14]. [15] applies Gaussian process regression(GPR) to the estimation of traffic frequencies within a public transport network. Their approach is not applicable to the problem of this work as pedestrian mobility patterns arise in the traffic flow due to the non-random but motivated individual behaviour, which was also result of the analysis of about 2'500 traces of train station visitors in [16]. More on challenges for pedestrian modelling can be found in [17]. [18] shows in a study of 210 infrastructure planning projects that the inaccuracy of traffic forecasts can be immense. In this paper we propose a new GPR based method to tackle the pedestrian quantity estimation problem which explores the prior knowledge of trajectory patterns.

3 GPR with Trajectory Patterns

In the paper we focus on the pedestrian quantity estimation in closed environments, e.g., train stations, shopping malls and zoos. Unlike the outdoor pedestrain quantity estimation, the continuous tracking technologies, e.g. global positioning system (GPS), are not feasible due to the lack of GPS signal in buildings and expensive deployment of the hardware. Recently developed technologies (lightbeams, video surveillance, Bluetooth meshes) record episodic movement data [19] or its location based aggregate, presence counts at low expenses. Episodic movement data is represented by tuples $< o, p, t >$ of moving object identifier o, discrete location identifier p and corresponding timestamp t. The location based aggregate, presence counts, for time interval Δt, as known as number of visits, quantity or traffic frequency, is defined as

$$NV(p, \Delta t) = | < o, p, t >, t \in \Delta t| \ . \tag{1}$$

To estimate the traffic volume at unmeasured locations, we propose a nonparametric Bayesian method, Gaussian process with a random-walk based trajectory kernel. The method explores not only the commonly used information in the literature, e.g. traffic network structures (retrieved by tessellation from the floor plan sketch) and recorded (or aggregated) presence counts NV at some measurement locations, but also the move preferences of pedestrians (trajectory patterns) collected from the local experts.

Consider a traffic network $\tilde{\mathcal{G}}(\tilde{\mathbf{V}}, \tilde{\mathbf{E}})$ with N vertices and M edges. For some of the edges, we observe the pedestrian quantities, denoted as $\mathbf{y} = \{y_s := NV(\tilde{e}_s, \Delta t) : s = 1, \ldots, S\}$. Additionally, we have the information of the major pedestrian movement patterns $\mathcal{T} = \{T_1, T_2, \ldots\}$ over the traffic network, collected from the local experts or the tracking technology (e.g. Bluetooth tracking technology). Obviously, taking into account the trajectory patterns is beneficial to predict the unknown pedestrian quantities: The edges included in a trajectory pattern appear to have similar pedestrian quantities. To meet the challenge, we propose a nonparametric Bayesian regression model with trajectory based kernels.

The pedestrian quantity estimation over traffic networks can be viewed as a link prediction problem, where the predicted quantities associated with links (edges) are continuous variables. In the literature of statistical relational learning [20,21], commonly used GP relational methods are to introduce a latent variable to each vertex, and the values of edges is therefore modeled as a function of latent variables of the involved vertices, e.g. [22,23]. Although these methods have the advantage that the problem size remains linear with the size of the vertices, it is difficult to find appropriate functions to encode the relationship between the variables of vertices and edges for different applications.

In the scenario of pedestrian quantity estimation, we directly model the edge-oriented quantities [5,6,15] within a Gaussian process regression framework. First, we convert the original network $\tilde{\mathcal{G}}(\tilde{\mathbf{V}}, \tilde{\mathbf{E}})$ to an edge graph $\mathcal{G}(\mathbf{V}, \mathbf{E})$ that represents the adjacencies between edges of $\tilde{\mathcal{G}}$. In the edge graph $\mathcal{G}$, each vertex $v_i \in \mathbf{V}$ is an edge of $\tilde{\mathcal{G}}$; and two vertices of $\mathcal{G}$ are connected if and only if their corresponding edges share a common endpoint in $\tilde{\mathcal{G}}$. To each vertex $v_i \in \mathbf{V}$ in the edge graph, we introduce a latent variable f_i which represents the true pedestrian quantity at v_i. It is value of a function over the edge graph and the known trajectory pattens, as well as the possible information about the features of the vertex. The observed pedestrian quantities (within a time interval Δt) are conditioned on the latent function values with Gaussian noise ϵ_i

$$y_i = f_i + \epsilon_i, \ \ \epsilon_i \sim \mathcal{N}(0, \sigma^2) \ . \tag{2}$$

As mathmatical form and parameters of the function are random and unknown, f_i is also unknown and random. For an infinite number of vertices, the function values $\{f_1, f_2, \ldots\}$ can be represented as an infinite dimensional vector. Within a nonparametric Bayesian framework, we assume that the infinite dimensional random vector follows a Gaussian process (GP) prior with mean function $m(x_i)$ and covariance function $k(x_i, x_j)$ [24]. In turn, any finite set of function values $\mathbf{f} = \{f_i : i = 1, ..., M\}$ has a multivariate Gaussian distribution with mean and covariances computed with the mean and covariance functions of the GP [24].

Without loss of generality, we assume zero mean so that the GP is completely specified by the covariance function. Formally, the multivariate Gaussian prior distribution of the function values $\mathbf{f}$ is written as

$$P(\mathbf{f}|\mathbf{X}) = \mathcal{N}(0, K),$$

where K denotes the $M \times M$ covariance matrix, whose ij-th entry is computed in terms of the covariance function. If there are vertex features $\mathbf{x} = \{x_1, ..., x_M\}$ available, e.g., the spatial representation of traffic edges, a typical choice for the covariance function is the squared exponential kernel with isotropic distance measure:

$$k(x_i, x_j) = \kappa^2 \exp\left(-\frac{\rho^2}{2}\sum_{d}^{D}(x_{i,d} - x_{j,d})^2\right), \tag{3}$$

where κ and ρ are hyperparameters.

Since the latent variables $\mathbf{f}$ are linked together into an edge graph $\mathcal{G}$, it is obvious that the covariances are closely related to the network structure: the variables are highly correlated if they are adjacent in $\mathcal{G}$, and vice versa. Therefore we can also employ graph kernels, e.g. the regularized Laplacian kernel, as the covariance functions:

$$K = \left[\beta(L + I/\alpha^2)\right]^{-1}, \tag{4}$$

where α and β are hyperparameters. L denotes the combinatorial Laplacian, which is computed as $L = D - A$, where A denotes the adjacency matrix of the graph $\mathcal{G}$. D is a diagonal matrix with entries $d_{i,i} = \sum_j A_{i,j}$.

Although graph kernels have some successful applications to public transportation networks [15], there are probably limitations when applying the network-based kernels to the scenario of closed environments: the pedestrians in a train station or a shopping mall have favorite or commonly used routes, they are not randomly distributed on the networks. In a train station, the pedestrian flow on the main corridor is most likely unrelated to that on the corridors leading to the offices, even if the corridors are adjacent. To incorporate the information of the move preferences (trajectory patterns, collected from the local experts or tracking technology) into the model, we explore a graph kernel inspired with the diffusion process [25].

Assume that a pedestrian randomly moves on the edge graph $\mathcal{G}$. From a vertex i he jumps to a vertex j with $n_{i,j}^k$ possible random walks of length k, where $n_{i,j}^k$ is equal to $[A^k]_{i,j}$. Intuitively, the similarity of two vertices is related to the number and the length of the random walks between them. Based on diffusion process, the similarity between vertices v_i and v_j is defined as

$$s(v_i, v_j) = \left[\sum_{k=1}^{\infty}\frac{\lambda^k}{k!}A^k\right]_{ij}, \tag{5}$$

where $0 \leq \lambda \leq 1$ is a hyperparameter. All possible random walks between v_i and v_j are taken into account in similarity computation, however the contributions of longer walks are discounted with a coefficient $\lambda^k/k!$. The similarity matrix is not always positive semi-definite. To get a valid kernel, the combinatorial Laplacian is used and the covariance matrix is defined as [25]:

$$K = \left[\sum_{k=1}^{\infty} \frac{\lambda^k}{k!} L^k \right] = \exp(\lambda L) \ . \tag{6}$$

On a traffic network within closed environment, the pedestrian will move not randomly, but with respect to a set of trajectory patterns and subpatterns denoted as sequences of vertices [26], e.g.,

$$\left\{ \begin{array}{l} T_1 = v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6, \\ T_2 = v_2 \rightarrow v_3 \rightarrow v_4, \\ T_3 = v_4 \rightarrow v_5 \rightarrow v_1, \\ \quad \cdots \end{array} \right\} . \tag{7}$$

Each trajectory pattern T_ℓ can also be represented as an adjacency matrix in which $\hat{A}_{i,j} = 1$ iff $v_i \rightarrow v_j \in T_\ell$ or $v_i \leftarrow v_j \in T_\ell$. The subpatterns are subsequences of the trajectories. For example, the subpatterns of T_1 are $\{v_1 \rightarrow v_3, v_3 \rightarrow v_5, v_5 \rightarrow v_6, v_1 \rightarrow v_3 \rightarrow v_5, v_3 \rightarrow v_5 \rightarrow v_6\}$. Given a set of trajectory patterns $\mathcal{T} = \{T_1, T_2, \ldots\}$, a random walk is valid and can be counted in similarity computation, if and only if all steps in the walk belong to $\mathcal{T}$ and subpatterns of $\mathcal{T}$. Thus we have

$$\begin{aligned} \hat{s}(v_i, v_j) &= \left[\sum_{k=1}^{\infty} \frac{\lambda^k}{k!} \hat{A}^k \right]_{ij}, & \hat{K} &= \left[\sum_{k=1}^{\infty} \frac{\lambda^k}{k!} \hat{L}^k \right] = \exp(\lambda \hat{L}) \\ \hat{A} &= \sum_{\ell} \hat{A}_\ell, & \hat{L} &= \hat{D} - \hat{A}, \end{aligned} \tag{8}$$

where $\hat{D}$ is a diagonal matrix with entries $\hat{d}_{i,i} = \sum_j \hat{A}_{i,j}$.

For pedestrian quantities $\mathbf{f}_u$ at unmeasured locations u, the predictive distribution can be computed as follows. Based on the property of GP, the observed and unobserved quantities $(\mathbf{y}, \mathbf{f}_u)^T$ follows a Gaussian distribution

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_u \end{bmatrix} \sim \mathcal{N} \left(0, \begin{bmatrix} \hat{K}_{\overline{u},\overline{u}} + \sigma^2 I & \hat{K}_{\overline{u},u} \\ \hat{K}_{u,\overline{u}} & \hat{K}_{u,u} \end{bmatrix} \right), \tag{9}$$

where $\hat{K}_{u,\overline{u}}$ is the corresponding entries of $\hat{K}$ between the unmeasured vertices u and measured ones $\overline{u}$. $\hat{K}_{\overline{u},\overline{u}}$, $\hat{K}_{u,u}$, and $\hat{K}_{\overline{u},u}$ are defined equivalently. I is an identity matrix of size $|\overline{u}|$. Finally the conditional distribution of the unobserved pedestrian quantities is still Gaussian with the mean m and the covariance matrix Σ:

$$\begin{aligned} m &= \hat{K}_{u,\overline{u}} (\hat{K}_{\overline{u},\overline{u}} + \sigma^2 I)^{-1} \, \mathbf{y} \\ \Sigma &= \hat{K}_{u,u} - \hat{K}_{u,\overline{u}} (\hat{K}_{\overline{u},\overline{u}} + \sigma^2 I)^{-1} \, \hat{K}_{\overline{u},u} \ . \end{aligned}$$

Besides pedestrian quantity estimation, incorporating trajectory patterns also enables effectively finding sensor placements that are most informative for traffic estimation on the whole network. To identify the most informative locations $\mathcal{I}$, we employ the exploration strategy, maximizing mutual information [27].

$$\arg\max_{\mathcal{I}\subset\mathbf{V}} H(\mathbf{V}\backslash\mathcal{I}) - H(\mathbf{V}\backslash\mathcal{I} \mid \mathcal{I}) \; . \tag{10}$$

It is equal to find a set of vertices $\mathcal{I}$, which maximally reduces the entropy of the traffic at the unmeasured locations $\mathbf{V}\backslash\mathcal{I}$. Since the entropy and the conditional entropy of Gaussian variables can be completely specified with covariances, the selection procedure is only based on covariances of vertices, not involves any pedestrian quantity observations. To solve the optimization problem, we employ a poly-time approximate method [27]. In particular, starting from an empty set $\mathcal{I} = \emptyset$, each vertex is selected with the criterion:

$$v_* \leftarrow \arg\max_{v\in\mathbf{V}\backslash\mathcal{I}} \; H_\epsilon(v \mid \mathcal{I}) - H_\epsilon(v \mid \overline{\mathcal{I}}) \; , \tag{11}$$

where $\overline{\mathcal{I}}$ denotes the vertex set $\mathbf{V}\backslash(\mathcal{I} \cup v)$. $H_\epsilon(x|Z) := H(x|Z')$ denotes an approximation of the entropy $H(x|Z)$, where any element z in $Z' \subset Z$ satisfies the constraint that the covariance between z and x is larger than a small value ϵ. Within the GP framework, the approximate entropy $H_\epsilon(x|Z)$ is computed as

$$\begin{aligned} H_\epsilon(x \mid Z) &= \frac{1}{2}\ln 2\pi e \sigma^2_{x|Z'} \\ \sigma^2_{x|Z'} &= \hat{K}_{x,x} - \hat{K}^T_{x,Z'}\hat{K}^{-1}_{Z',Z'}\hat{K}_{x,Z'} \; . \end{aligned} \tag{12}$$

The term $\hat{K}_{x,Z'}$ is the corresponding entries of $\hat{K}$ between the vertex x and a set of vertices Z'. $\hat{K}_{x,x}$ and $\hat{K}_{Z',Z'}$ are defined equivalently. Given the informative trajectory pattern kernel, the pedestrian quantity observations at the vertices selected with the criterion (11) can well estimate the situation of the whole network. Sec. 4.3 shows a successful application to the zoo of Duisburg data.

4 Experimental Analysis

Our intention here is to investigate the following questions: (Q1) Can the proposed method integrate with expert knowledge on preferred movement patterns in closed environments and thus improve the prediction accuracy on pedestrian quantity estimation? (Q2) Can the proposed method choose sensor locations to better monitor pedestrian quantities in an industrial scenario? To this aim, we evaluate the method on two datasets: synthetic German train station pedestrian data and real-world episodic movement data collected with Bluetooth tracking technology at the zoo of Duisburg (Germany). We compare our method with state-of-the-art traffic volume estimation approaches. As discussed in Section 2, kNN methods are extensively used for traffic volume estimation [5]. The latest version of this approach, the *Spatial kNN* [7], has many successful applications and is considered as a baseline. In the experiments we use distance-weighted 5-Nearest Neighbors. Particularly this method detects for each unmeasured edge the 5 nearest neighbors among the measured ones. Their traffic frequencies are weighted by distance to achieve prediction for the unmeasured ones. As this is a geometric algorithm which requires spatial representation of the traffic network, we apply Fruchterman Reingold [28] algorithm to lay out the test networks in two-dimensional space and achieve spatial representations. Distances between edges are computed with Euclidian metric. Additionally, we compare our method to GPR with commonly used kernels, including regularized Laplacian (RL), squared exponential (SE) and diffusion kernel (Diff). The prediction performance of the methods is measured with mean absolute error (MAE) $MAE = n^{-1} \sum_{i=1}^{n} |y_i - f_i|$.

4.1 German Train Station Data

To approximate the true situation, we study traffic networks of 170 largest public train stations in Germany, an example shown as Fig. 2. The distributions of vertex-degree and vertex-number are visualized in Fig. 3. Given the collected information of the real-world train stations, the synthetic data is generated as follows. We apply the real vertex-degree distribution to the random network generator described in [29] and draw the train station like random graphs of order 10. In these graphs we generate pedestrian flows between dead ends (vertices of degree one), as no pedestrians permanently stay in a train station. The dead ends are selected pairwise and edge frequencies are sampled along the shortest connecting path with a random frequency of maximal 10,000 persons, which is a reasonable approximation for train station traffic networks. Afterwards we select a random set of edges (ranging from 10 to 50 percent of all edges) as monitored locations. Traffic frequencies at these edges are viewed as evidence to estimate frequencies at unmeasured ones. At each setting, we repeat the experiment 100 times and report the distributions of prediction performance for each method.

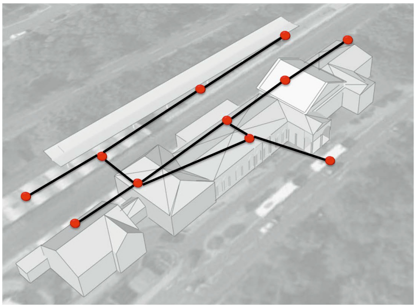


Fig. 2. Sketch of train station Hofheim (Germany) with traffic network overlay

Experimental results are depicted in Fig. 4. Grouped in blocks are the different experiment configurations (different number of monitored edges). Statistics

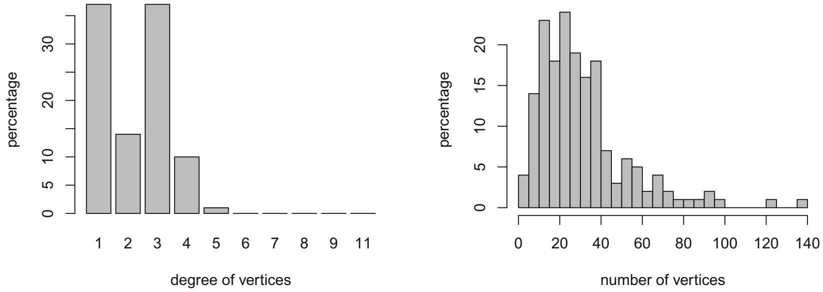


Fig. 3. Distributions of vertex-degree (right) and number of vertices (left) among 170 large German train stations

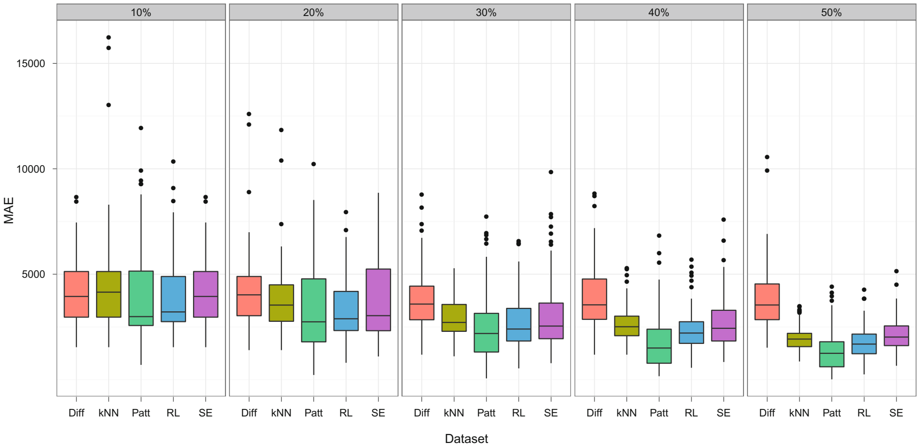


Fig. 4. Pedestrian quantity estimation on networks of train stations. Performance is measured by MAE at settings with different ratios of monitored edges (10 to 50 percent from left to right). The five methods: GPR with diffusion kernel (Diff), spatial k-nearest neighbor (kNN), GPR with trajectory pattern kernel (Patt), GPR with regularized Laplacian (RL) and GPR with squared exponential kernel (SE).

on the MAE distribution per method are depicted in the five boxplots. Throughout the tests, our method achieved minimal MAE and minimal average MAE, and therefore best results for the pedestrian quantity estimation problem. The proposed method outperformed commonly used kNN approach, especially when traffic networks are sparsely monitored. With increasing the number of monitored edges, all methods, except the GPR with diffusion kernel, provide better performance on pedestrian quantity estimation given that MAE decreased and did not scatter that much. Within the GP framework, the proposed trajectory pattern kernel achieved best performance compared to other kernels.

4.2 Zoo of Duisburg Data

We apply the proposed method to a real world dataset of visitor movement in the zoo of Duisburg (Germany). The dataset consists of episodic movement data [19] and was collected with a mesh of 15 Bluetooth scanners [3,30] (see map in Fig. 5). Within a period of 7 days (07/26/11–08/02/11) all Bluetooth enabled devices (smartphones or intercoms) were scanned and log-entries attached to the log-file. Thus, the dataset consists of tuples (device identifier, timestamp, location). In order to perform the tests, the traffic network is build from the sensor positions. Each sensor becomes a vertex. To achieve ground truth for the traffic

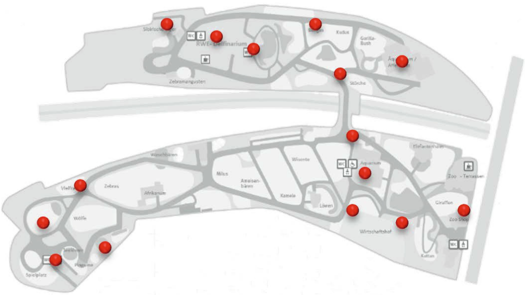


Fig. 5. The network of the zoo in Duisburg (Germany) and the positions of the 15 Bluetooth scanners

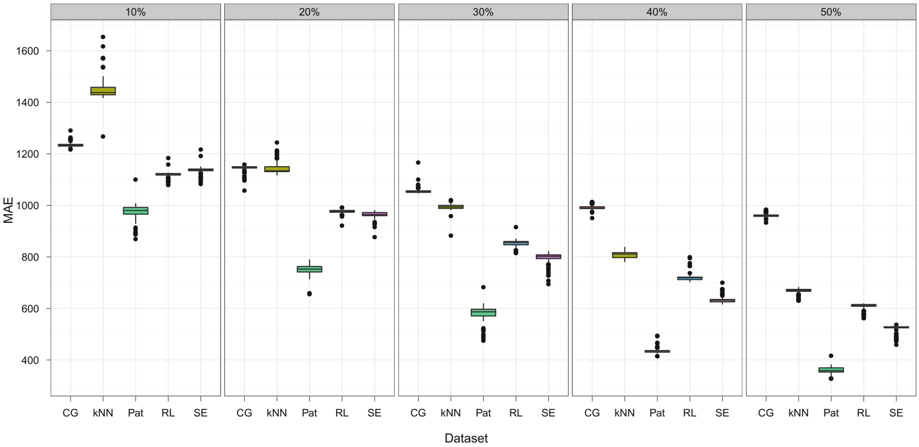


Fig. 6. Pedestrian quantity estimation on network of the zoo in Duisburg (Germany). Performance is measured by MAE at settings with different ratios of monitored edges (10 to 50 percent from left to right). The five methods: GPR with diffusion kernel (Diff), spatial k-nearest neighbor (kNN), GPR with trajectory pattern kernel (Patt), GPR with regularized Laplacian (RL) and GPR with squared exponential kernel (SE)

volume prediction, temporal aggregates of recorded transitions between sensors, as proposed in [19], become scaled by the Bluetooth representativity (in this case at the zoo approximately 6 percent). Due to the uncertainties in episodic movement data transitions in the dataset are not limited to neighbouring sensor positions, but occur between arbitrary pairs of sensors. In our case this results in a traffic network consisting of 102 edges and 15 vertices. The recorded trajectories of the zoo visitors become the trajectory pattern input to the *trajectory pattern kernel*. Similar to the previously synthetically generated data, the real world experiments are conducted with different percentages of measured edges. Measurement edges are chosen uniformly at random 100 times for each dataset.

As shown in Figure 6 on the experimental results, the proposed method again achieved the best prediction performance for the pedestrian quantity estimation problem in comparison to other state-of-the-art methods. Incorporating expert knowledge on movement preferences allows for the model to well capture the dependencies of traffic at different edges and, in turn, to improve prediction accuracy.

4.3 Sensor Placement with Trajectory Patterns

Besides the traffic volume estimation, another interesting task is to give a solution to the question where to place the sensors such that the traffic over the whole network can be well estimated. Based on the proposed trajectory pattern kernel, we perform the sensor placement procedure on the zoo of Duisburg data.

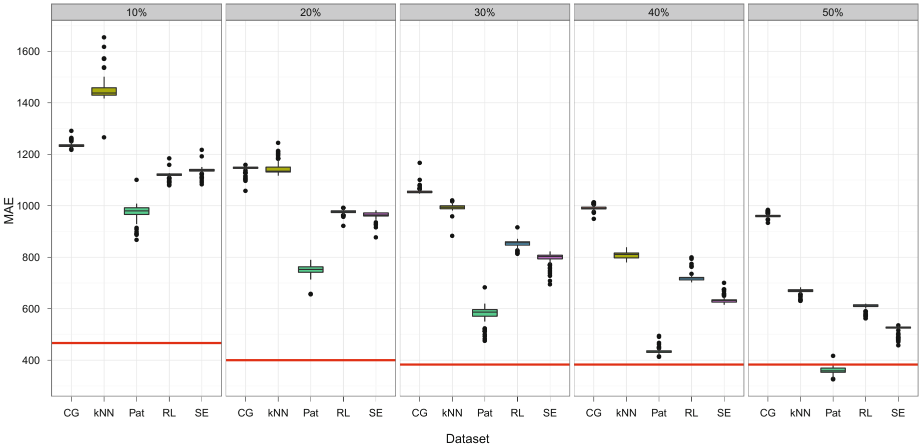


Fig. 7. Traffic Flow Estimation performance measured by MAE for 5 real world datasets with different ratios of known edges (10 to 50 percent) and five methods: GPR with Diffusion kernel (Diff), Spatial k-Nearest Neighbor (kNN), GPR with the proposed Trajectory Pattern kernel (Patt), GPR with Regularized Laplacian (RL) and GPR with Squared Exponential (SE) in comparison to (Patt) with mutual information based sensor placement (horizontal line).

Afterwards, pedestrian quantity estimation based on resulting sensor placement is carried out and performance is measured with MAE. The red horizontal line in Fig. 7 depicts the sensor placement performances in comparison to previous random placement. For sparse sensor distribution (low percentages of measurement edges), the sensor placement has a high positive impact on the prediction performance. However, for higher sensor numbers the random placement may outperform the mutual information based sensor placement. One reason is that this placement is not optimal but near optimal. Another possible explanation is given by the data. Due to noise or other unexpected anomalies in the data which are not consistent to the prior knowledge on trajectory patterns.

5 Conclusions and Future Work

Pedestrian volume estimation is an important problem for mobility analysis in many application scenarios such as emergency support systems, quality-of-service evaluation and billboard placement, risk analysis and location ranking. This work proposed a nonparametric Bayesian method to tackle the pedestrian quantity estimation problem which explores the expert knowledge of trajectory patterns. We validated our proposed method on two datasets: synthetic German train station pedestrian data and real-world dataset collected with of Bluetooth tracking technology at the zoo of Duisburg. Furthermore, we addressed the question for sensor placement in an industrial scenario with the trajectory based graph kernel. The empirical analysis demonstrated that incorporating trajectory patterns can largely improve the traffic prediction accuracy in comparison to other state-of-the-art methods. Our work also provides an efficient and applicable solution to pedestrian volume estimation in industrial real world scenarios.

This work focussed on pedestrian volume estimation in closed environments (zoo, train station, terminal, etc.) because in closed environments different methods can be studied and compared under controlled circumstances. For instance, movements in these closed environments are not influenced by residing persons or unexpected pedestrian sinks or sources like tram stops, living houses, etc. Nevertheless, our proposed approach was not based on this assumption and future work should validate performance on arbitrary traffic networks. Another future research direction is to focus on temporal aspects of pedestrian movement and the creation of time dynamic models using at once dynamic expert knowledge and dynamic measurements. Also combination of measurements and expert knowledge at heterogeneous spatial granularities is promising for industrial applications (e.g. combination of (1) movement patterns among dedicated points of interest retrieved from social media and (2) pedestrian counts from video surveillance on (3) a city center traffic network). This question is of high interest in near future, as valuable (episodic) data on people's movement is expected to become widely available e.g. by billing data, logfiles on social media usage or wireless communication networks (GSM, WLAN, Bluetooth) [19].

Acknowledgements. This work was supported by the European Project Emergency Support System (ESS 217951) and the Fraunhofer ATTRACT Fellowship STREAM. We gratefully thank the zoo of Duisburg for supporting data collection.

References

1. Liebig, T., Körner, C., May, M.: Scalable sparse bayesian network learning for spatial applications. In: ICDM Workshops, pp. 420–425. IEEE Computer Society (2008)
2. Liebig, T., Körner, C., May, M.: Fast visual trajectory analysis using spatial bayesian networks. In: ICDM Workshops, pp. 668–673. IEEE Computer Society (2009)
3. Bruno, R., Delmastro, F.: Design and Analysis of a Bluetooth-Based Indoor Localization System. In: Conti, M., Giordano, S., Gregori, E., Olariu, S. (eds.) PWC 2003. LNCS, vol. 2775, pp. 711–725. Springer, Heidelberg (2003)
4. Zhao, F., Park, N.: Using geographically weighted regression models to estimate annual average daily traffic. Journal of the Transportation Research Board 1879(12), 99–107 (2004)
5. Gong, X., Wang, F.: Three improvements on knn-npr for traffic flow forecasting. In: Proceedings of the 5th International Conference on Intelligent Transportation Systems, pp. 736–740. IEEE Press (2002)
6. Lam, W.H.K., Tang, Y.F., Tam, M.: Comparison of two non-parametric models for daily traffic forecasting in hong kong. Journal of Forecasting 25(3), 173–192 (2006)
7. May, M., Hecker, D., Körner, C., Scheider, S., Schulz, D.: A vector-geometry based spatial knn-algorithm for traffic frequency predictions. In: Data Mining Workshops, International Conference on Data Mining, pp. 442–447. IEEE Computer Society, Los Alamitos (2008)
8. Scheider, S., May, M.: A method for inductive estimation of public transport traffic using spatial network characteristics. In: Proceedings of the 10th AGILE International Conference on Geographic Information Sciences, pp. 1–8. Aalborg University (2007)
9. May, M., Scheider, S., Rösler, R., Schulz, D., Hecker, D.: Pedestrian flow prediction in extensive road networks using biased observational data. In: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2008, p. 67. ACM, New York (2008)
10. Wang, X., Kockelmann, K.M.: Forecasting network data: Spatial interpolation of traffic counts from texas data. Journal of the Transportation Research Board 2105(13), 100–108 (2009)
11. Liebig, T., Xu, Z.: Pedestrian monitoring system for indoor billboard evaluation. Journal of Applied Operational Research 4(1), 28–36 (2012)
12. Liebig, T.: A general pedestrian movement model for the evaluation of mixed indoor-outdoor poster campaigns. In: Proc. of the Third International Conference on Applied Operation Research - ICAOR 2011, pp. 289–300. Tadbir Operational Research Group Ltd. (2011)
13. Liebig, T., Stange, H., Hecker, D., May, M., Körner, C., Hofmann, U.: A general pedestrian movement model for the evaluation of mixed indoor-outdoor poster campaigns. In: Proc. of the Third International Workshop on Pervasive Advertising and Shopping (2010)

14. Kretz, T.: Pedestrian traffic: on the quickest path. Journal of Statistical Mechanics: Theory and Experiment P03012 (March 2009)
15. Neumann, M., Kersting, K., Xu, Z., Schulz, D.: Stacked gaussian process learning. In: Proceeding of the 9th IEEE International Conference on Data Mining (ICDM 2009), pp. 387–396. IEEE Computer Society (2009)
16. Li, M., Konomi, S., Sezaki, K.: Understanding and modeling pedestrian mobility of train-station scenarios. In: Sabharwal, A., Karrer, R., Zhong, L. (eds.) WINTECH, pp. 95–96. ACM (2008)
17. Harney, D.: Pedestrian modelling: current methods and future directions. Road Transport Research 11(4), 38–48 (2002)
18. Flyvbjerg, B., Skamris, H., Mette, K., Buhl, S.L.: Inaccuracy in traffic forecasts. Transport Reviews 26(1), 1–24 (2006)
19. Andrienko, N., Andrienko, G., Stange, H., Liebig, T., Hecker, D.: Visual Analytics for Understanding Spatial Situations from Episodic Movement Data. KI - Künstliche Intelligenz, 1–11 (2012)
20. De Raedt, L.: Logical and Relational Learning. Springer (2008)
21. Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning. The MIT Press (2007)
22. Yu, K., Chu, W., Yu, S., Tresp, V., Xu, Z.: Stochastic relational models for discriminative link prediction. In: Neural Information Processing Systems (2006)
23. Chu, W., Sindhwani, V., Ghahramani, Z., Keerthi, S.: Relational learning with gaussian processes. In: Neural Information Processing Systems (2006)
24. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press (2006)
25. Kondor, R.I., Lafferty, J.D.: Diffusion kernels on graphs and other discrete input spaces. In: Proceeding of the International Conference on Machine Learning, pp. 315–322 (2002)
26. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14. IEEE Computer Society, Washington, DC (1995)
27. Krause, A., Guestrin, C., Gupta, A., Kleinberg, J.: Near-optimal sensor placements: maximizing information while minimizing communication cost. In: Proceedings of the 5th International Conference on Information Processing in Sensor Networks, IPSN 2006, pp. 2–10. ACM, New York (2006)
28. Fruchterman, T.M.J., Reingold, E.M.: Graph Drawing by Force-directed Placement. Software: Practice and Experience 21(11), 1129–1164 (1991)
29. Viger, F., Latapy, M.: Efficient and Simple Generation of Random Simple Connected Graphs with Prescribed Degree Sequence. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 440–449. Springer, Heidelberg (2005)
30. Hallberg, J., Nilsson, M., Synnes, K.: Positioning with Bluetooth. In: 10th International Conference on Telecommunications, vol. 2, pp. 954–958 (2003)

Socioscope: Spatio-temporal Signal Recovery from Social Media

Jun-Ming Xu[1], Aniruddha Bhargava[2], Robert Nowak[2], and Xiaojin Zhu[1,2]

[1] Department of Computer Sciences
[2] Department of Electrical and Computer Engineering
University of Wisconsin-Madison, Madison WI 53706, USA
{xujm,jerryzhu}@cs.wisc.edu, aniruddha@wisc.edu, nowak@ece.wisc.edu

Abstract. Many real-world phenomena can be represented by a spatio-temporal signal: where, when, and how much. Social media is a tantalizing data source for those who wish to monitor such signals. Unlike most prior work, we assume that the target phenomenon is known and we are given a method to count its occurrences in social media. However, counting is plagued by sample bias, incomplete data, and, paradoxically, data scarcity – issues inadequately addressed by prior work. We formulate signal recovery as a Poisson point process estimation problem. We explicitly incorporate human population bias, time delays and spatial distortions, and spatio-temporal regularization into the model to address the noisy count issues. We present an efficient optimization algorithm and discuss its theoretical properties. We show that our model is more accurate than commonly-used baselines. Finally, we present a case study on wildlife roadkill monitoring, where our model produces qualitatively convincing results.

1 Introduction

Many real-world phenomena of interest to science are spatio-temporal in nature. They can be characterized by a real-valued intensity function $\mathbf{f} \in \mathbb{R}_{\geq 0}$, where the value $f_{s,t}$ quantifies the prevalence of the phenomenon at location s and time t. Examples include wildlife mortality, algal blooms, hail damage, and seismic intensity. Direct instrumental sensing of $\mathbf{f}$ is often difficult and expensive. Social media offers a unique sensing opportunity for such spatio-temporal signals, where users serve the role of "sensors" by posting their experiences of a target phenomenon. For instance, social media users readily post their encounters with dead animals: *"I saw a dead crow on its back in the middle of the road."*

There are at least three challenges faced when using human social media users as sensors:

1. Social media sources are not always reliable and consistent, due to factors including the vagaries of language and the psychology of users. This makes identifying topics of interest and labeling social media posts extremely challenging.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 644–659, 2012.

2. Social media users are not under our control. In most cases, users cannot be directed or focused or maneuvered as we wish. The distribution of human users (our sensors) depends on many factors unrelated to the sensing task at hand.
3. Location and time stamps associated with social media posts may be erroneous or missing. Most posts do not include GPS coordinates, and self-reported locations can be inaccurate or false. Furthermore, there can be random delays between an event of interest and the time of the social media post related to the event.

Most prior work in social media event analysis has focused on the first challenge. Sophisticated natural language processing techniques have been used to identify social media posts relevant to a topic of interest [21,2,16] and advanced machine learning tools have been proposed to discover popular or emerging topics in social media [1,12,22]. We discuss the related work in detail in Section 3.

Our work in this paper focuses on the latter two challenges. We are interested in a specific topic or target phenomenon of interest that is given and fixed beforehand, and we assume that we are also given a (perhaps imperfect) method, such as a trained text classifier, to identify target posts. The first challenge is relevant here, but is not the focus of our work. The main concerns of this paper are to deal with the highly non-uniform distribution of human users (sensors), which profoundly affects our capabilities for sensing natural phenomena such as wildlife mortality, and to cope with the uncertainties in the location and time stamps associated with related social media posts. The main contribution of the paper is robust methodology for deriving accurate spatiotemporal maps of the target phenomenon in light of these two challenges.

2 The Socioscope

We propose Socioscope, a probabilistic model that robustly recovers spatiotemporal signals from social media data. Formally, consider $\mathbf{f}$ defined on discrete spatiotemporal bins. For example, a bin (s, t) could be a U.S. state s on day t, or a county s in hour t. From the first stage we obtain $x_{s,t}$, the count of target social media posts within that bin. The task is to estimate $f_{s,t}$ from $x_{s,t}$. A commonly-used estimate is $\widehat{f_{s,t}} = x_{s,t}$ itself. This estimate can be justified as the maximum likelihood estimate of a Poisson model $\mathbf{x} \sim \text{Poisson}(\mathbf{f})$. This idea underlines several emerging systems such as earthquake damage monitoring from Twitter [8]. However, this estimate is unsatisfactory since the counts $x_{s,t}$ can be *noisy*: as mentioned before, the estimate ignores population bias – more target posts are generated when and where there are more social media users; the location of a target post is frequently inaccurate or missing, making it difficult to assign to the correct bin; and target posts can be quite sparse even though the total volume of social media is huge. Socioscope addresses these issues.

For notational simplicity, we often denote our signal of interest by a vector $\mathbf{f} = (f_1, \dots, f_n)^\top \in \mathbb{R}^n_{\geq 0}$, where f_j is a non-negative target phenomenon intensity in *source bin* $j = 1 \dots n$. We will use a wildlife example throughout the section. In this example, a source bin is a spatiotemporal unit such as "California, day 1," and f_j is the squirrel activity level in that unit. The mapping between index j and the aforementioned (s, t) is one-one and will be clear from context.

2.1 Correcting Human Population Bias

For now, assume each target post comes with precise location and time meta data. This allows us to count x_j, the number of target posts in bin j. Given x_j, it is tempting to use the maximum likelihood estimate $\widehat{f_j} = x_j$ which assumes a simple Poisson model $x_j \sim \text{Poisson}(f_j)$. However, this model is too naive: Even if $f_j = f_k$, e.g., the level of squirrel activities is the same in two bins, we would expect $x_j > x_k$ if there are more people in bin j than in bin k, simply because more people see the same group of squirrels.

To account for this population bias, we define an "active social media user population intensity" (loosely called "human population" below) $\mathbf{g} = (g_1, \dots, g_n)^\top \in \mathbb{R}^n_{\geq 0}$. Let z_j be the count of *all* social media posts in bin j, the vast majority of which are not about the target phenomenon. We assume $z_j \sim \text{Poisson}(g_j)$. Since typically $z_j \gg 0$, the maximum likelihood estimate $\widehat{g_j} = z_j$ is reasonable.

Importantly, we then posit the Poisson model

$$x_j \sim \text{Poisson}(\eta(f_j, g_j)). \tag{1}$$

The intensity is defined by a *link function* $\eta(f_j, g_j)$. In this paper, we simply define $\eta(f_j, g_j) = f_j \cdot g_j$ but note that other more sophisticated link functions can be learned from data. Given x_j and z_j, one can then easily estimate f_j with the plug-in estimator $\widehat{f_j} = x_j / z_j$.

2.2 Handling Noisy and Incomplete Data

This would have been the end of the story if we could reliably assign each post to a source bin. Unfortunately, this is often not the case for social media. In this paper, we focus on the problem of spatial uncertainty due to noisy or incomplete social media data. A prime example of spatial uncertainty is the lack of location meta data in posts from Twitter (called tweets).[1] In recent data we collected, only 3% of tweets contain the latitude and longitude at which they were created. Another 47% contain a valid user self-declared location in his or her profile (e.g., "New York, NY"). However, such location does not automatically change while the user travels and thus may not be the true location at which a tweet is posted.

[1] It may be possible to recover occasional location information from the tweet text itself instead of the meta data, but the problem still exists.

The remaining 50% do not contain location at all. Clearly, we cannot reliably assign the latter two kinds of tweets to a spatiotemporal source bin. [2]

To address this issue, we borrow an idea from Positron Emission Tomography [19]. In particular, we define m *detector bins* which are conceptually distinct from the n source bins. The idea is that an event originating in some source bin goes through a transition process and ends up in one of the detector bins, where it is detected. This transition is modeled by an $m \times n$ matrix $\mathbf{P} = \{P_{ij}\}$ where

$$P_{ij} = \Pr(\text{detector } i \mid \text{source } j). \tag{2}$$

$\mathbf{P}$ is column stochastic: $\sum_{i=1}^{m} P_{ij} = 1, \forall j$. We defer the discussion of our specific $\mathbf{P}$ to a case study, but we mention that it is possible to reliably estimate $\mathbf{P}$ directly from social media data (more on this later). Recall the target post intensity at source bin j is $\eta(f_j, g_j)$. We use the transition matrix to define the target post intensity h_i (note that h_i can itself be viewed as a link function $\tilde{\eta}(\mathbf{f}, \mathbf{g})$) at detector bin i:

$$h_i = \sum_{j=1}^{n} P_{ij} \eta(f_j, g_j). \tag{3}$$

For the spatial uncertainty that we consider, we create three kinds of detector bins. For a source bin j such as "California, day 1," the first kind collects target posts on day 1 whose latitude and longitude meta data is in California. The second kind collects target posts on day 1 without latitude and longitude meta data, but whose user self-declared profile location is in California. The third kind collects target posts on day 1 without any location information. Note the third kind of detector bin is shared by all other source bins for day 1, such as "Nevada, day 1," too. Consequently, if we had $n = 50T$ source bins corresponding to the 50 US states over T days, there would be $m = (2 \times 50 + 1)T$ detector bins.

Critically, our observed target counts $\mathbf{x}$ are now with respect to the m detector bins instead of the n source bins: $\mathbf{x} = (x_1, \ldots, x_m)^\top$. We will also denote the count sub-vector for the first kind of detector bins by $\mathbf{x}^{(1)}$, the second kind $\mathbf{x}^{(2)}$, and the third kind $\mathbf{x}^{(3)}$. The same is true for the overall counts $\mathbf{z}$. A trivial approach is to only utilize $\mathbf{x}^{(1)}$ and $\mathbf{z}^{(1)}$ to arrive at the plug-in estimator

$$\widehat{f_j} = x_j^{(1)} / z_j^{(1)}. \tag{4}$$

As we will show, we can obtain a better estimator by incorporating noisy data $\mathbf{x}^{(2)}$ and incomplete data $\mathbf{x}^{(3)}$. $\mathbf{z}^{(1)}$ is sufficiently large and we will simply ignore $\mathbf{z}^{(2)}$ and $\mathbf{z}^{(3)}$.

[2] Another kind of spatiotemporal uncertainty exists in social media even when the local and time meta data of every post is known: social media users may not immediately post right at the spot where a target phenomenon happens. Instead, there usually is an unknown time delay and spatial shift between the phenomenon and the post generation. For example, one may not post a squirrel encounter on the road until she arrives at home later; the local and time meta data only reflects tweet-generation at home. This type of spatiotemporal uncertainty can be addressed by the same source-detector transition model.

2.3 Socioscope: Penalized Poisson Likelihood Model

We observe target post counts $\mathbf{x} = (x_1, \ldots, x_m)$ in the detector bins. These are modeled as independently Poisson distributed random variables:

$$x_i \sim \text{Poisson}(h_i), \text{ for } i = 1 \ldots m. \tag{5}$$

The log likelihood factors as

$$\ell(\mathbf{f}) = \log \prod_{i=1}^{m} \frac{h_i^{x_i} e^{-h_i}}{x_i!} = \sum_{i=1}^{m} (x_i \log h_i - h_i) + c, \tag{6}$$

where c is a constant. In (6) we treat g as given.

Target posts may be scarce in some detector bins. Indeed, we often have zero target posts for the wildlife case study to be discussed later. This problem can be mitigated by the fact that many real-world phenomena are spatiotemporally smooth, i.e., "neighboring" source bins in space or time tend to have similar intensity. We therefore adopt a penalized likelihood approach by constructing a graph-based regularizer. The undirected graph is constructed so that the nodes are the source bins. Let $\mathbf{W}$ be the $n \times n$ symmetric non-negative weight matrix. The edge weights are such that W_{jk} is large if j and k correspond to neighboring bins in space and time. Since $\mathbf{W}$ is domain specific, we defer its construction to the case study.

Before discussing the regularizer, we need to perform a change of variables. Poisson intensity $\mathbf{f}$ is non-negative, necessitating a constrained optimization problem. It is more convenient to work with an unconstrained problem. To this end, we work with the exponential family natural parameters of Poisson. Specifically, let

$$\theta_j = \log f_j, \quad \psi_j = \log g_j. \tag{7}$$

Our specific link function becomes $\eta(\theta_j, \psi_j) = e^{\theta_j + \psi_j}$. The detector bin intensities become $h_i = \sum_{j=1}^{n} P_{ij} \eta(\theta_j, \psi_j)$.

Our graph-based regularizer applies to θ directly:

$$\Omega(\theta) = \frac{1}{2} \theta^\top \mathbf{L} \theta, \tag{8}$$

where $\mathbf{L}$ is the combinatorial graph Laplacian [5]: $\mathbf{L} = \mathbf{D} - \mathbf{W}$, and $\mathbf{D}$ is the diagonal degree matrix with $D_{jj} = \sum_{k=1}^{n} W_{jk}$.

Finally, Socioscope is the following penalized likelihood optimization problem:

$$\min_{\theta \in \mathbb{R}^n} - \sum_{i=1}^{m} (x_i \log h_i - h_i) + \lambda \Omega(\theta), \tag{9}$$

where λ is a positive regularization weight.

2.4 Optimization

We solve the Socioscope optimization problem (9) with BFGS, a quasi-Newton method [14]. The gradient can be easily computed as

$$\nabla = \lambda \mathbf{L}\theta - \mathbf{H}\mathbf{P}^\top(\mathbf{r} - 1), \tag{10}$$

where $\mathbf{r} = (r_1 \dots r_m)$ is a ratio vector with $r_i = x_i/h_i$, and $\mathbf{H}$ is a diagonal matrix with $H_{jj} = \eta(\theta_j, \psi_j)$.

We initialize θ with the following heuristic. Given counts $\mathbf{x}$ and the transition matrix P, we compute the least-squared projection η_0 to $\|\mathbf{x} - \mathbf{P}\eta_0\|_2$. This projection is easy to compute. However, η_0 may contain negative components not suitable for Poisson intensity. We force positivity by setting $\eta_0 \leftarrow \max(10^{-4}, \eta_0)$ element-wise, where the floor 10^{-4} ensures that $\log \eta_0 > -\infty$. From the definition $\eta(\theta, \psi) = \exp(\theta + \psi)$, we then obtain the initial parameter

$$\theta_0 = \log \eta_0 - \psi. \tag{11}$$

Our optimization is efficient: problems with more than one thousand variables (n) are solved in about 15 seconds with fminunc() in Matlab.

2.5 Parameter Tuning

The choice of the regularization parameter λ has a profound effect on the smoothness of the estimates. It may be possible to select these parameters based on prior knowledge in certain problems, but for our experiments we select these parameters using a cross-validation (CV) procedure, which gives us a fully data-based and objective approach to regularization.

CV is quite simple to implement in the Poisson setting. A hold-out set of data can be constructed by simply sub-sampling events from the total observation uniformly at random. This produces a partial data set of a subset of the counts that follows precisely the same distribution as the whole set, modulo a decrease in the total intensity per the level of subsampling. The complement of the hold-out set is what remains of the full dataset, and we will call this the training set. The hold-out set is taken to be a specific fraction of the total. For theoretical reasons beyond the scope of this paper, we do not recommend leave-one-out CV [18,6].

CV is implemented by generating a number of random splits of this type (we can generate as many as we wish), and for each split we run the optimization algorithm above on the training set for a range of values of λ. Then compute the (unregularized) value of the log-likelihood on the hold-out set. This provides us with an estimate of the log-likelihood for each setting of λ. We simply select the setting that maximizes the estimated log-likelihood.

2.6 Theoretical Considerations

The natural measure of signal-to-noise in this problem is the number of counts in each bin. The higher the counts, the more stable and "less noisy" our estimators will be. Indeed, if we directly observe $x_i \sim \text{Poisson}(h_i)$, then the normalized error $\mathbf{E}[(\frac{x_i - h_i}{h_i})^2] = h_i^{-1} \approx x_i^{-1}$. So larger counts, due to larger underlying intensities, lead to small errors on a relative scale. However, the accuracy of our recovery also depends on the regularity of the underlying function f. If it is very smooth, for example a constant function, then the error would be inversely proportional to the total number of counts, not the number in each individual bin. This is because in the extreme smooth case, f is determined by a single constant.

To give some insight into dependence of the estimate on the total number of counts, suppose that f is the underlying continuous intensity function of interest. Furthermore, let f be a Hölder α-smooth function. The parameter α is related to the number of continuous derivatives f has. Larger values of α correspond to smoother functions. Such a model is reasonable for the application at hand, as discussed in our motivation for regularization above. We recall the following minimax lower bound, which follows from the results in [7,20].

Theorem 1. *Let f be a Hölder α-smooth d-dimensional intensity function and suppose we observe N events from the distribution Poisson(f). Then there exists a constant $C_\alpha > 0$ such that*

$$\inf_{\widehat{f}} \sup_{f} \frac{\mathbf{E}[\|\widehat{f} - f\|_1^2]}{\|f\|_1^2} \;\geq\; C_\alpha N^{\frac{-2\alpha}{2\alpha+d}} \;,$$

where the infimum is over all possible estimators. The error is measured with the 1-norm, rather than two norm, which is a more appropriate and natural norm in density and intensity estimation. The theorem tells us that no estimator can achieve a faster rate of error decay than the bound above. There exist many types of estimators that nearly achieve this bound (e.g., to within a log factor), and with more work it is possible to show that our regularized estimators, with adaptively chosen bin sizes and appropriate regularization parameter settings, could also nearly achieve this rate. For the purposes of this discussion, the lower bound, which certainly applies to our situation, will suffice.

For example, consider just two spatial dimensions ($d = 2$) and $\alpha = 1$ which corresponds to Lipschitz smooth functions, a very mild regularity assumption. Then the bound says that the error is proportional to $N^{-1/2}$. This gives useful insight into the minimal data requirements of our methods. It tells us, for example, that if we want to reduce the error of the estimator by a factor of say 2, then the total number of counts must be increased by a factor of 4. If the smoothness α is very large, then doubling the counts can halve the error. The message is simple. More events and higher counts will provide more accurate estimates.

3 Related Work

To our knowledge, there is no comparable prior work that focuses on robust single recovery from social media (i.e., the "second stage" as we mentioned in the introduction). However, there has been considerable related work on the first stage, which we summarize below.

Topic detection and tracking (TDT) aims at identifying emerging topics from text stream and grouping documents based on their topics. The early work in this direction began with news text streamed from newswire and transcribed from other media [1]. Recent research focused on user-generated content on the web and on the spatio-temporal variation of topics. Latent Dirichlet Allocation (LDA) [3] is a popular unsupervised method to detect topics. Mei *et al.* [12] extended LDA by taking spatio-temporal context into account to identify subtopics from weblogs. They analyzed the spatio-temporal pattern of topic θ by $\Pr(time|\theta, location)$ and $\Pr(location|\theta, time)$, and showed that documents created from the same spatio-temporal context tend to share topics. In the same spirit, Yin *et al.* [22] studied GPS-associated documents, whose coordinates are generated by Gaussian Mixture Model in their generative framework. Cataldi *et al.* [4] proposed a *feature-pivot* method. They first identified keywords whose occurrences dramatically increase in a specified time interval and then connected the keywords to detect emerging topics. Besides text, social network structure also provides important information for detecting community-based topics and user interests.

Event detection is highly related to TDT. Yang *et al.* [21] uses clustering algorithm to identify events from news streams. Others tried to distinguish posts related to real world events from posts about non-events, such as describing daily life or emotions [2]. Real world events were also detected in Flickr photos with meta information and Twitter. Other researchers were interested in events with special characteristics, such as controversial events and local events. Sakaki *et al.* [16] monitored Twitter to detect real-time events such as earthquakes and hurricanes.

Another line of related work uses social media as a data source to answer scientific questions [11]. Most previous work studied questions in linguistic, sociology and human interactions. For example, Eisenstein *et al.* [9] studied the geographic linguistic variation with geotagged social media. Gupte *et al.* [10] studied social hierarchy and stratification in online social network.

As stated earlier, Socioscope differs from past work in its focus on robust signal recovery on predefined target phenomena. The target posts may be generated at a very low, though sustained, rate, and are corrupted by noise. The above approaches are unlikely to estimate the underlying intensity accurately.

4 A Synthetic Experiment

We start with a synthetic experiment whose known ground-truth intensity $\mathbf{f}$ allows us to quantitatively evaluate the effectiveness of Socioscope. The synthetic experiment matches the case study in the next section. There are 48 US continental states plus Washington DC, and $T = 24$ hours. This leads to a total of $n = 1176$ source bins, and $m = (2 \times 49 + 1)T = 2376$ detector bins. The transition matrix $\mathbf{P}$ is the same as in the case study, to be discussed later. The overall counts $\mathbf{z}$ are obtained from actual Twitter data and $\widehat{\mathbf{g}} = \mathbf{z}^{(1)}$.

We design the ground-truth target signal $\mathbf{f}$ to be temporally constant but spatially varying. Figure 1(a) shows the ground-truth $\mathbf{f}$ spatially. It is a mixture of two Gaussian distributions discretized at the state level. The modes are in Washington and New York, respectively. From $\mathbf{P}$, $\mathbf{f}$ and $\mathbf{g}$, we generate the observed target post counts for each detector bin by a Poisson random number generator: $x_i \sim \text{Poisson}(\sum_{j=1}^{n} P_{i,j} f_j g_j)$, $i = 1 \ldots m$. The sum of counts in $\mathbf{x}^{(1)}$ is 56, in $\mathbf{x}^{(2)}$ 1106, and in $\mathbf{x}^{(3)}$ 1030.

Table 1. Relative error of different estimators

(i) scaled $\mathbf{x}^{(1)}$	14.11
(ii) scaled $\mathbf{x}^{(1)}/\mathbf{z}^{(1)}$	46.73
(iii) Socioscope with $\mathbf{x}^{(1)}$	0.17
(iv) Socioscope with $\mathbf{x}^{(1)} + \mathbf{x}^{(2)}$	1.83
(v) Socioscope with $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$	0.16
(vi) **Socioscope with $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$**	**0.12**

Given $\mathbf{x}, \mathbf{P}, \mathbf{g}$, We compare the relative error $\|\mathbf{f} - \hat{\mathbf{f}}\|^2 / \|\mathbf{f}\|^2$ of several estimators in Table 1:

(i) $\hat{\mathbf{f}} = \mathbf{x}^{(1)} / (\epsilon_1 \sum \mathbf{z}^{(1)})$, where ϵ_1 is the fraction of tweets with precise location stamp (discussed later in case study). Scaling matches it to the other estimators. Figure 1(b) shows this simple estimator, aggregated spatially. It is a poor estimator: besides being non-smooth, it contains 32 “holes” (states with zero intensity, colored in blue) due to data scarcity.

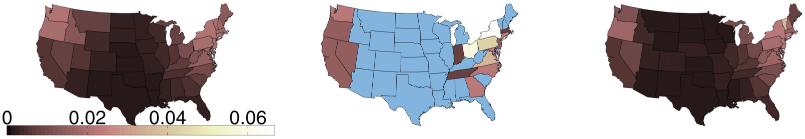
(ii) $\hat{\mathbf{f}} = \mathbf{x}_j^{(1)} / (\epsilon_1 \mathbf{z}_j^{(1)})$ which naively corrects the population bias as discussed in (4). It is even worse than the simple estimator, because naive bin-wise correction magnifies the variance in sparse $\mathbf{x}^{(1)}$.

(iii) Socioscope with $\mathbf{x}^{(1)}$ only. This simulates the practice of discarding noisy or incomplete data, but regularizing for smoothness. The relative error was reduced dramatically.

(iv) Same as (iii) but replace the values of $\mathbf{x}^{(1)}$ with $\mathbf{x}^{(1)} + \mathbf{x}^{(2)}$. This simulates the practice of ignoring the noise in $\mathbf{x}^{(2)}$ and pretending it is precise. The result is worse than (iii), indicating that simply including noisy data may hurt the estimation.

(v) Socioscope with $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ separately, where $\mathbf{x}^{(2)}$ is treated as noisy by $\mathbf{P}$. It reduces the relative error further, and demonstrates the benefits of treating noisy data specially.

(vi) Socioscope with the full $\mathbf{x}$. It achieves the lowest relative error among all methods, and is the closest to the ground truth (Figure 1(c)). Compared to (v), this demonstrates that even counts $\mathbf{x}^{(3)}$ without location can also help us to recover $\mathbf{f}$ better.



(a) ground-truth $\mathbf{f}$ (b) scaled $\mathbf{x}^{(1)}$ (c) Socioscope

Fig. 1. The synthetic experiment

5 Case Study: Roadkill

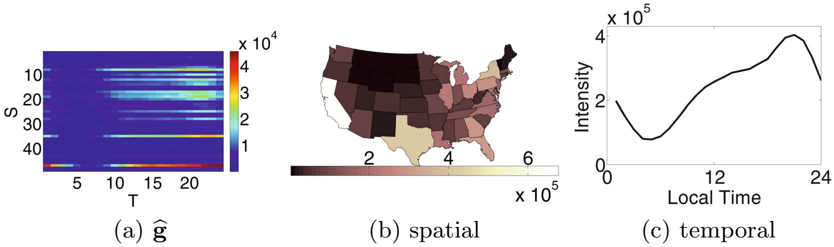
We were unaware of public benchmark data sets to test robust signal recovery from social media (the "second stage"). Several social media datasets were released recently, such as the ICWSM data challenges and the TREC microblog track. These datasets were intended to study trending "hot topics" such as the Arabic Spring, Olympic Games, or presidential elections. They are not suitable for low intensity sustained target phenomena which is the focus of our approach. In particular, these datasets do not contain ground-truth spatio-temporal intensities and are thus not appropriate testbeds for the problems we are trying to address. Instead, we report a real-world case study on the spatio-temporal intensity of roadkill for several common wildlife species from Twitter posts.

The study of roadkill has values in ecology, conservation, and transportation safety. The target phenomenon consists of roadkill events for a specific species within the continental United States during September 22–November 30, 2011. Our spatio-temporal source bins are state×hour-of-day. Let s index the 48 continental US states plus District of Columbia. We aggregate the 10-week study period into 24 hours of a day. The target counts $\mathbf{x}$ are still sparse even with aggregation: for example, most state-hour combination have zero counts for armadillo and the largest count in $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ is 3. Therefore, recovering the underlying signal $\mathbf{f}$ remains a challenge. Let t index the hours from 1 to 24. This results in $|s| = 49, |t| = 24, n = |s||t| = 1176, m = (2|s| + 1)|t| = 2376$. We will often index source or detector bins by the subscript (s,t), in addition to i or j, below. The translation should be obvious.

5.1 Data Preparation

We chose Twitter as our data source because public tweets can be easily collected through its APIs. All tweets include time meta data. However, most tweets do not contain location meta data, as discussed earlier.

Overall Counts $\mathbf{z}^{(1)}$ and Human Population Intensity $\mathbf{g}$. To obtain the overall counts $\mathbf{z}$, we collected tweets through the Twitter stream API using bounding boxes covering continental US. The API supplied a subsample of *all*



(a) $\widehat{\mathbf{g}}$ (b) spatial (c) temporal

Fig. 2. Human population intensity $\widehat{\mathbf{g}}$

tweets (not just target posts) with geo-tag. Therefore, all these tweets include precise latitude and longitude on where they were created. Through a reverse geocoding database (`http://www.datasciencetoolkit.org`), we mapped the coordinates to a US state. There are a large number of such tweets. Counting the number of tweets in each state-hour bin gave us $\mathbf{z}^{(1)}$, from which $\mathbf{g}$ is estimated.

Figure 2 shows the estimated $\widehat{\mathbf{g}}$. The x-axis is hour of day and y-axis is the states, ordered by longitude from east (top) to west (bottom). Although $\widehat{\mathbf{g}}$ in this matrix form contains full information, it can be hard to interpret. Therefore, we visualize aggregated results as well: First, we aggregate out time in $\widehat{\mathbf{g}}$: for each state s, we compute $\sum_{t=1}^{24} \widehat{g_{s,t}}$ and show the resulting intensity maps in Figure 2(b). Second, we aggregate out state in $\widehat{\mathbf{g}}$: for each hour of day t, we compute $\sum_{s=1}^{49} \widehat{g_{s,t}}$ and show the daily curve in Figure 2(c). From these two plots, we clearly see that human population intensity varies greatly both spatially and temporally.

Identifying Target Posts to Obtain Counts x. To produce the target counts $\mathbf{x}$, we need to first identify target posts describing roadkill events. Although not part of Socioscope, we detail this preprocessing step here for reproducibility.

In step 1, we collected tweets using a keyword API. Each tweet must contain the wildlife name (e.g., "squirrel(s)") *and* the phrase "ran over". We obtained 5857 squirrel tweets, 325 chipmunk tweets, 180 opossum tweets and 159 armadillo tweets during the study period. However, many such tweets did not actually describe roadkill events. For example, *"I almost ran over an armadillo on my longboard, luckily my cat-like reflexes saved me."* Clearly, the author did not kill the armadillo.

In step 2, we built a binary text classifier to identify target posts among them. Following [17], the tweets were case-folded without any stemming or stopword removal. Any user mentions preceded by a "@" were replaced by the anonymized user name "@USERNAME". Any URLs staring with "http" were replaced by the token "HTTPLINK". Hashtags (compound words following "#") were not split and were treated as a single token. Emoticons, such as ":)" or ":D", were also included as tokens. Each tweet is then represented by a feature vector consisting of unigram and bigram counts. If any unigram or bigram included animal names, we added an additional feature by replacing the animal name with the generic token “ANIMAL”. For example, we would created an extra feature “over ANIMAL” for the bigram “over raccoon”. The training data consists of 1,450 manually labeled tweets in August 2011 (i.e., *outside* our study period). These training tweets contain hundreds of animal species, not just the target species. The binary label is whether the tweet is a true first-hand roadkill experience. We trained a linear Support Vector Machine (SVM). The CV accuracy is nearly 90%. We then applied this SVM to classify tweets surviving step 1. Those tweets receiving a positive label were treated as target posts.

In step 3, we produce $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}$ counts. Because these target tweets were collected by the keyword API, the nature of the Twitter API means that most do not contain precise location information. As mentioned earlier, only 3% of them contain coordinates. We processed this 3% by the same reverse geocoding database to map them to a US state s, and place them in the $x_{s,t}^{(1)}$ detection bins. 47% of the target posts do not contain coordinates but can be mapped to a US state from user self-declared profile location. These are placed in the $x_{s,t}^{(2)}$ detection bins. The remaining 50% contained no location meta data, and were placed in the $x_t^{(3)}$ detection bins. [3]

Constructing the Transition Matrix P. In this study, $\mathbf{P}$ characterizes the fraction of tweets which were actually generated in source bin (s,t) end up in the three detector bins: precise location $st^{(1)}$, potentially noisy location $st^{(2)}$, and missing location $t^{(3)}$. We define $\mathbf{P}$ as follows:

$P_{(s,t)^{(1)},(s,t)} = 0.03$, and $P_{(r,t)^{(1)},(s,t)} = 0$ for $\forall r \neq s$ to reflect the fact that we know precisely 3% of the target posts' location.

$P_{(r,t)^{(2)},(s,t)} = 0.47 M_{r,s}$ for all r, s. M is a 49×49 “mis-self-declare” matrix. $M_{r,s}$ is the probability that a user self-declares in her profile that she is in state r, but her post is in fact generated in state s. We estimated $\mathbf{M}$ from a separate large set of tweets with both coordinates and self-declared profile locations. The $\mathbf{M}$ matrix is asymmetric and interesting in its own right: many posts self-declared in California or New York were actually produced all over the country; many self-declared in Washington DC were actually produced in Maryland or Virgina; more posts self-declare Wisconsin but were actually in Illinois than the other way around.

$P_{t^{(3)},(s,t)} = 0.50$. This aggregates tweets with missing information into the third kind of detector bins.

Specifying the Graph Regularizer. Our graph has two kinds of edges. Temporal edges connect source bins with the same state and adjacent hours by weight w_t. Spatial edges connect source bins with the same hour and adjacent states by weight w_s. The regularization weight λ was absorbed into w_t and w_s. We tuned the weights w_t and w_s with CV on the 2D grid $\{10^{-3}, 10^{-2.5}, \ldots, 10^{3}\}^2$.

[3] There were actually only a fraction of all tweets without location which came from all over the world. We estimated this US/World fraction using $\mathbf{z}$.

5.2 Results

We present results on four animals: armadillos, chipmunks, squirrels, opossums. Perhaps surprisingly, precise roadkill intensities for these animals are apparently unknown to science (This serves as a good example of the value Socioscope may provide to wildlife scientists). Instead, domain experts were only able to provide a range map of each animal, see the left column in Figure 3. These maps indicate presence/absence only, and were extracted from NatureServe [15]. In addition, the experts defined armadillo and opossum as nocturnal, chipmunk as diurnal, and squirrels as both crepuscular (active primarily during twilight) and diurnal. Due to the lack of quantitative ground-truth, our comparison will necessarily be qualitative in nature.

Socioscope provides sensible estimates on these animals. For example, Figure 4(a) shows counts $\mathbf{x}^{(1)} + \mathbf{x}^{(2)}$ for chipmunks which is very sparse (the largest count in any bin is 3), and Figure 4(b) the Socioscope estimate $\widehat{\mathbf{f}}$. The axes are the same as in Figure 2(a). In addition, we present the state-by-state intensity maps in the middle column of Figure 3 by aggregating $\widehat{\mathbf{f}}$ spatially. The Socioscope results match the range maps well for all animals. The right column in Figure 3 shows the daily animal activities by aggregating $\widehat{\mathbf{f}}$ temporally. These curves match the animals' diurnal patterns well, too.

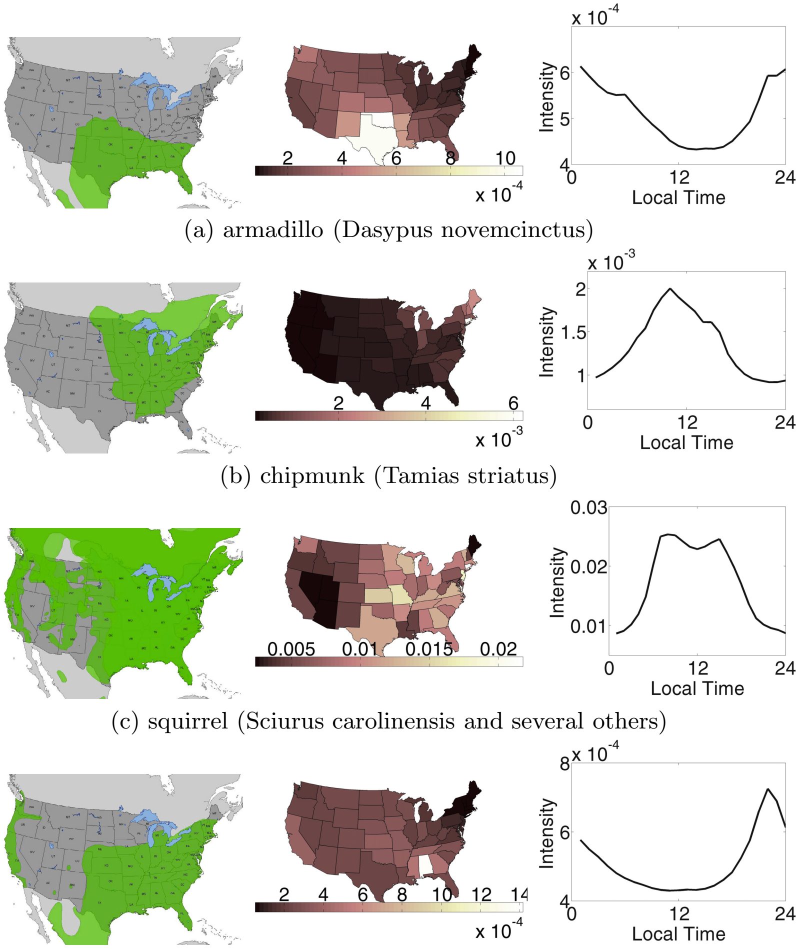
The Socioscope estimates are superior to the baseline methods in Table 1. Due to space limit we only present two examples on chipmunks, but note that similar observations exist for all animals. The baseline estimator of simply scaling $\mathbf{x}^{(1)} + \mathbf{x}^{(2)}$ produced the temporal and spatial aggregates in Figure 5(a,b). Compared to Figure 3(b, right), the temporal curve has a spurious peak around 4-5pm. The spatial map contains spurious intensity in California and Texas, states outside the chipmunk range as shown in Figure 3(b, left). Both are produced by population bias when and where there were strong background social media activities (see Figure 2(b,c)). In addition, the spatial map contains 27 "holes" (states with zero intensity, colored in blue) due to data scarcity. In contrast, Socioscope's estimates in Figure 3 avoid this problem by regularization. Another baseline estimator $(\mathbf{x}^{(1)} + \mathbf{x}^{(2)})/\mathbf{z}^{(1)}$ is shown in Figure 5(c). Although corrected for population bias, this estimator lacks the transition model and regularization. It does not address data scarcity either.

6 Future Work

Using social media as a data source for spatio-temporal signal recovery is an emerging area. Socioscope represents a first step toward this goal. There are many open questions:

1. We treated target posts as certain. In reality, a natural language processing system can often supply a confidence. For example, a tweet might be deemed to be a target post only with probability 0.8. It will be interesting to study ways to incorporate such confidence into our framework.

2. The temporal delay and spatial displacement between the target event and the generation of a post is commonplace, as discussed in footnote 2. Estimating



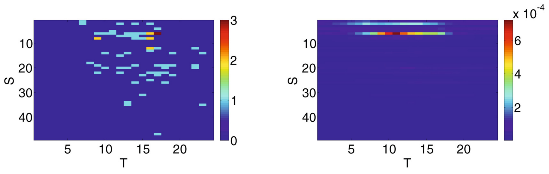
(a) armadillo (Dasypus novemcinctus)

(b) chipmunk (Tamias striatus)

(c) squirrel (Sciurus carolinensis and several others)

(d) opossum (Didelphis virginiana)

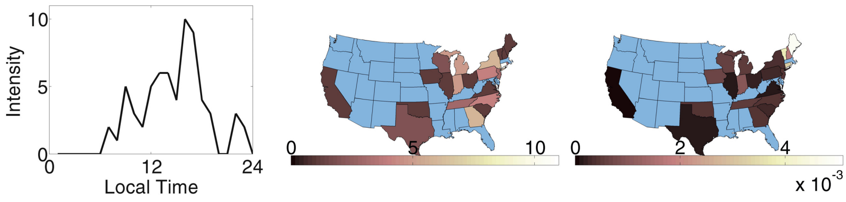
Fig. 3. Socioscope estimates match animal habits well. (Left) range map from NatureServe, (Middle) Socioscope $\widehat{\mathbf{f}}$ aggregated spatially, (Right) $\widehat{\mathbf{f}}$ aggregated temporally.



(a) $\mathbf{x}^{(1)} + \mathbf{x}^{(2)}$

(b) Socioscope $\widehat{\mathbf{f}}$

Fig. 4. Raw counts and Socioscope $\widehat{\mathbf{f}}$ for chipmunks



(a) $\mathbf{x}^{(1)} + \mathbf{x}^{(2)}$ (b) $\mathbf{x}^{(1)} + \mathbf{x}^{(2)}$ (c) $(\mathbf{x}^{(1)} + \mathbf{x}^{(2)})/\mathbf{z}^{(1)}$

Fig. 5. Examples of inferior baseline estimators. In all plots, states with zero counts are colored in blue.

an appropriate transition matrix **P** from social media data so that Socioscope can handle such "point spread functions" remains future work.

3. It might be necessary to include psychology factors to better model the human "sensors." For instance, a person may not bother to tweet about a chipmunk roadkill, but may be eager to do so upon seeing a moose roadkill.

4. Instead of discretizing space and time into bins, one may adopt a spatial point process model to learn a continuous intensity function instead [13].

Addressing these considerations will further improve Socioscope.

Acknowledgments. We thank Megan K. Hines from Wildlife Data Integration Network for providing range maps and guidance on wildlife. This work is supported in part by the Global Health Institute at the University of Wisconsin-Madison.

References

1. Allan, J.: Topic Detection and Tracking: Event-Based Information Organization. Kluwer Academic Publishers, Norwell (2002)
2. Becker, H., Mor, N., Gravano, L.: Beyond trending topics: Real-world event identification on twitter. In: Proceedings of the 15th International AAAI Conference on Weblogs and Social Media, pp. 438–441 (2011)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)
4. Cataldi, M., Di Caro, L., Schifanella, C.: Emerging topic detection on twitter based on temporal and social terms evaluation. In: Proceedings of the 10th International Workshop on Multimedia Data Mining, pp. 4:1–4:10 (2010)
5. Chung, F.R.K.: Spectral graph theory. In: Regional Conference Series in Mathematics. American Mathematical Society, Providence (1997)
6. Cornec, M.: Concentration inequalities of the cross-validation estimate for stable predictors. Arxiv preprint arXiv:1011.5133 (2010)
7. Donoho, D., Johnstone, I., Kerkyacharian, G., Picard, D.: Density estimation by wavelet thresholding. The Annals of Statistics 24, 508–539 (1996)
8. Earle, P., Guy, M., Buckmaster, R., Ostrum, C., Horvath, S., Vaughan, A.: OMG earthquake! Can Twitter improve earthquake response? Seismological Research Letters 81(2), 246–251 (2010)

9. Eisenstein, J., O'Connor, B., Smith, N.A., Xing, E.P.: A latent variable model for geographic lexical variation. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 1277–1287 (2010)
10. Gupte, M., Shankar, P., Li, J., Muthukrishnan, S., Iftode, L.: Finding hierarchy in directed online social networks. In: Proceedings of the 20th International Conference on World Wide Web, pp. 557–566 (2011)
11. Lazer, D., Pentland, A.S., Adamic, L., Aral, S., Barabasi, A.L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., Alstyne, M.V.: Life in the network: the coming age of computational social science. Science 323(5915), 721–723 (2009)
12. Mei, Q., Liu, C., Su, H., Zhai, C.: A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In: Proceedings of the 15th International Conference on World Wide Web, pp. 533–542 (2006)
13. Møller, J., Waagepetersen, R.: Statistical inference and simulation for spatial point processes. Monographs on statistics and applied probability. Chapman & Hall/CRC, Boca Raton (2004)
14. Nocedal, J., Wright, S.: Numerical optimization. Springer series in operations research. Springer, New York (1999)
15. Patterson, B.D., Ceballos, G., Sechrest, W., Tognelli, M.F., Brooks, T., Luna, L., Ortega, P., Salazar, I., Young, B.E.: Digital distribution maps of the mammals of the western hemisphere, version 3.0. Tech. rep., NatureServe, Arlington, VA (2007), http://www.natureserve.org/
16. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 851–860 (2010)
17. Settles, B.: Closing the Loop: Fast, Interactive Semi-Supervised Annotation With-Queries on Features and Instances. In: Proceedings of the Conference on Empirical-Methods in Natural Language Processing, Edinburgh, UK, pp. 1467–1478 (2011)
18. Van Der Laan, M., Dudoit, S.: Unified cross-validation methodology for selectionamong estimators and a general cross-validated adaptive epsilon-net estimator:Finite sample oracle inequalities and examples. U.C. Berkeley Division of Biostatistics Working Paper Series, pp. 130–236 (2003)
19. Vardi, Y., Shepp, L.A., Kaufman, L.: A statistical model for positron emission tomography. Journal of the American Statistical Association 80(389), 8–37 (1985)
20. Willett, R., Nowak, R.: Multiscale poisson intensity and density estimation. IEEE Transactions on Information Theory 53(9), 3171–3187 (2007)
21. Yang, Y., Pierce, T., Carbonell, J.: A study of retrospective and on-line event detection. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 28–36 (1998)
22. Yin, Z., Cao, L., Han, J., Zhai, C., Huang, T.: Geographical topic discovery and comparison. In: Proceedings of the 20th International Conference on World Wide Web, pp. 247–256 (2011)

A Framework for Evaluating the Smoothness of Data-Mining Results

Gaurav Misra, Behzad Golshan, and Evimaria Terzi

Boston University, Computer Science Department
{gm,behzad,evimaria}@cs.bu.edu

Abstract. The data-mining literature is rich in problems that are formalized as combinatorial-optimization problems. An indicative example is the *entity-selection* formulation that has been used to model the problem of selecting a subset of representative reviews from a review corpus [11,22] or important nodes in a social network [10]. Existing combinatorial algorithms for solving such entity-selection problems identify a set of entities (e.g., reviews or nodes) as important. Here, we consider the following question: how do small or large changes in the input dataset change the value or the structure of the such reported solutions?

We answer this question by developing a general framework for evaluating the *smoothness* (i.e, consistency) of the data-mining results obtained for the input dataset X. We do so by comparing these results with the results obtained for datasets that are within a small or a large distance from X. The algorithms we design allow us to perform such comparisons effectively and thus, approximate the results' smoothness efficiently. Our experimental evaluation on real datasets demonstrates the efficacy and the practical utility of our framework in a wide range of applications.

1 Introduction

Given a collection of reviews about a product, which are the most valuable reviews in a collection? In a social network of users, which are the most influential nodes in the network?

Many of such *entity-selection* problems (where the term entities is used as an abstraction for reviews and node networks) have been formalized in the data-mining literature as combinatorial-optimization problems; e.g., using coverage [11,22] or influence-maximization objectives [10]. Such formulations are characterized by a *solution space* and an *objective function*. The former defines the types of solutions the data analyst is interested in, e.g., a set of reviews or nodes. The latter provides a means for evaluating the *goodness* or *value* of every solution for a given dataset. In this context, the combinatorial methods output a single solution from the solution space – the one that optimizes (or approximates) the objective function.

While existing work focuses on designing efficient algorithms for finding (or approximating) this optimal solution, we claim that these algorithms fail to quantify how the *nature of the input dataset* affects the reported solutions. For

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 660–675, 2012.

example, if small changes in the input graph alter significantly the reported set of influential nodes, then any social study relying on existing algorithms would be highly untrustworthy. Similarly, if the set of influential nodes reported in the input dataset are identified as influential in *any* random dataset, then again the reported solution is insignificant.

In this paper, we propose a framework that allows us to quantify the relationship between the combinatorial solutions obtained by existing algorithms to the *nature of the input dataset*. More specifically, we propose the *smoothness measure*, which quantifies how the value and the structure of the reported solutions is altered by small (or large) changes in the input dataset. Related to ours is the work on statistical-significance testing of data-mining results using empirical p-values [5,17,16,23]. However, the empirical p-values encode the probability that there exists a random dataset with solution with (almost) identical value to the solution obtained for the original dataset. On the other hand, the smoothness is the expected similarity between the solutions of the sampled and the original datasets. The focus on the expected similarity of the solutions, allows us to sample datasets either randomly (as in the case of p-values) or within specified neighborhoods around the original data. This flexibility allows us to quantify not only the statistical significance of the obtained solutions, but also the variation of the solution within these small neighborhoods.

Another key characteristic of our work is that it departs from the narrow characterization of solutions using only the objective function. In particular, we view the solutions as combinatorial objects that have both value and structure. After all, the objective function is only a proxy for the analyst's intuition of what constitutes a good solution. In the case of the entity-selection problem the structure of the solution is determined by the actual elements that are selected to the reported set. Therefore, when comparing solutions not only do we compare their values, but also the actual entities they contain.

Apart from the proposition of the smoothness framework, which to the best of our knowledge is new, we also present a new efficient algorithm for sampling datasets that are within a certain distance from the original dataset. This algorithm is a key component to our algorithmic solution for approximating the smoothness of the data-mining results for entity-selection problems. Our experimental results (presented in Section 4) validate the utility and the practical utility of our framework as well as our algorithms in a wide range of entity-selection problems. Some indicative ones are the selection of representative set of reviews, authors within communities as well as nodes in social networks.

Although we present the application of our framework to entity-selection problems, we point out that our framework is general and can be used to evaluate the smoothness of all data-mining results – as long as the data-mining problem has been formalized as a combinatorial optimization problem.

The rest of the paper is organized as follows: first, we describe our framework in Section 2. In Section 3 we describe our algorithms for sampling datasets and in Section 4 we present experiments that demonstrate the utility of our methods. After reviewing the related work in Section 5, we conclude the paper in Section 6.

2 Framework Overview

In this section, we provide a generic description of our framework that takes as input a dataset, an objective function, and an algorithm for optimizing it, and reports how the solution reported by the algorithm is affected by small or large changes in the input dataset.

2.1 Optimization Problems

Every combinatorial-optimization problem consists of the following components: the input dataset X, the designated solution space $\mathscr{L}$and the objective function F, which maps every solution $\mathcal{S} \in \mathscr{L}$ to a real number $F(X, \mathcal{S})$. Henceforth, we use tuple $\langle X, \mathscr{L}, F\rangle$ to represent a combinatorial optimization problem, and $\mathcal{S}_X^*$ to represent the optimal solution for input dataset X. Here, we focus on maximization problems, where $\mathcal{S}_X^*$ is the solution for which $F(X, \mathcal{S})$ is maximized:

$$\mathcal{S}_X^* = \text{argmax}_{\mathcal{S}\in\mathscr{L}} F(X, \mathcal{S}).$$

We will use $\mathcal{S}^*$ instead of $\mathcal{S}_X^*$ whenever the dataset X is clear from the context.

Note that finding $\mathcal{S}^*$ might be **NP**-hard for some problems. In these cases, different heuristics and approximation algorithms are used to obtain a suboptimal solution. We will use $\mathcal{S}^*$ to denote the solution (optimal or not) produced by the algorithm chosen to solve the problem.

Among all data-mining problems that have been formalized using this framework are the following two, which we consider in this paper: the review-selection and the node-selection problems. Consider a collection of reviews for a particular product. This product has a set of features that can be commented in a review. For instance, the features of an mp3-player are: battery, sound quality, ability to record, and etc. Each reviewer may post a review that only covers a subset of these features (say only battery and sound quality). The goal of the review-selection problem is to pick k reviews that cover the maximum number of distinct features of the product.

Similarly, in the node-selection problem, the goal is to pick a set of k important nodes from a given network. That is, to select k nodes that have the highest influence over the entire network. Of course, in order to define such nodes we need to rely on an *information propagation* model. Given such a model, information propagates from active to inactive nodes; this propagation is usually probabilistic. Using an information propagation model, the important nodes are the ones that upon activation, will lead to maximum number of (expected) active nodes in the network. There are many applications for this problem. For instance, in a given graph of social relations, the selected nodes can be used to obtain an optimal advertising strategy.

We collectively refer to the review and the node-selection problems as *entity-selection problems*. The former one has been formalized using the MaxCov formulation [11,22], and the latter, using the MaxInfluence formulation [10]. We describe these formulations next.

$\textsc{MaxCov}(k)$: Given a universe of items U and m subsets of it ($\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$), the goal of $\textsc{MaxCov}$ problem is to pick k elements from $\mathcal{C}$ so that the number of distinct items that they cover from U is maximized. Formally, we would like to pick $\mathcal{S} \subseteq \mathcal{C}$ with $|\mathcal{S}| = k$ such that

$$F\text{-Cov}(\mathcal{S}) = \left| \bigcup_{C \in \mathcal{S}} C \right| \tag{1}$$

is maximized. In the case of review selection, the elements of $\mathcal{C}$ are the reviews and the universe consists of the product's features.

In an instance of $\textsc{MaxCov}$ problem $\langle X, \mathscr{L}, F \rangle$, the input dataset X is the set of universe subsets $\mathcal{C}$. $\mathscr{L}$ is the solution space and contains all valid solutions like $\mathcal{S}$ where $\mathcal{S}$ is the set of k selected subsets ($\mathcal{S} \subseteq \mathcal{C}$). Finally, F is the coverage objective function given by Equation (1).

The $\textsc{MaxCov}$ problem is known to be **NP**-hard. However, a greedy method of picking the subset with the largest number of uncovered elements at each step, is an $(1 - \frac{1}{e})$-approximation algorithm for the $\textsc{MaxCov}$ problem. Henceforth, the output of the mentioned greedy algorithm is what we refer to as $\mathcal{S}^*$ for the $\textsc{MaxCov}$ problem.

Observe that input dataset X, can be represented by a binary matrix in the following way: each column corresponds to an element in the universe U, and each row corresponds to one of the given subsets. The binary matrix has a value of 1 in position (i, j) *iff* the subset C_i contains the j-th element of U. Throughout, we consider X to be the mentioned binary matrix to keep the notation simple.

$\textsc{MaxInfluence}(K)$: Given a graph $G = (V, E)$ in which all the nodes are inactive, the goal is to find the best k nodes to activate, so that the expected number of active node after propagation would be maximized. Formally, we would like to pick $\mathcal{S} \subseteq V$ with $\mathcal{S} = k$ such that the following objective function would be maximized:

$$F\text{-Influence}(\mathcal{S}) = \text{ expected number of active nodes.}$$

The computations of the expected number of active nodes depends on the propagation model. In our study, we focus on the *independent cascade* (IC) model [10].[1]

In an instance of $\textsc{MaxInfluence}$ problem $\langle X, \mathscr{L}, F \rangle$, X is the input graph $G = (V, E)$. $\mathscr{L}$ is the set of all possible solutions where each solution $S \subseteq V$ and $|S| = k$. Finally, The objective function F is the F-Influence function.

Solving the $\textsc{MaxInfluence}$ problem is also **NP**-hard. However, a greedy algorithm that at every step picks the node with the largest marginal increase in the objective function again achieves an $(1 - \frac{1}{e})$-approximation for the objective. Henceforth, the output of this greedy algorithm is what we refer to as $\mathcal{S}^*$ for the $\textsc{MaxInfluence}$ problem.

Similar to $\textsc{MaxCov}$ problem, the input dataset X can be represented using a binary matrix. Since the input dataset is a graph, X can be the adjacency matrix that describes the input graph.

[1] Our results carry over to other propagation models.

2.2 Evaluating Data-Mining Results

Here we introduce *smoothness* and show how it can be adjusted to reveal different aspects of a given combinatorial optimization-problem.

We formally, define the smoothness of a combinatorial optimization-problem $\langle X, \mathscr{L}, F \rangle$ using the following formula:

$$\textsc{Smoothness}(X) = \sum_{X' \in \mathcal{X}} Pr(X') Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'})$$

The formula consists of three main parts: the *dataspace* $\mathcal{X}$, the *sampling probability distribution* $Pr(X')$, and the *similarity function* $Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'})$. The rest of this section describes these three components.

Dataspace. The dataspace $\mathcal{X}$ is the set of all datasets which share a *structural characteristic* with X. Intuitively, $\mathcal{X}$ is the collection of datasets which compare with X. So far, we have shown that the input dataset for both MaxCov and MaxInfluence problems is a binary matrix. As a result, we introduce two different possible choices of $\mathcal{X}$ for binary matrices: the *exchange dataspace* $\mathcal{X}_E$ and the *swap dataspace* $\mathcal{X}_W$. The former is the set of all datasets with the same number of ones as X. The latter contains the datasets that have the same row and column marginals as X. Finally, we point out that for each dataspace $\mathcal{X}$, there is a natural distance function $D(Y, Z)$ which returns the distance between datasets Y and Z in $\mathcal{X}$. Here, we use the following distance function to measure the distance between two binary matrices:

$$D(Y, Z) = |\{(i, j) | Y_{i,j} = 1 \wedge Z_{i,j} = 0\}| \tag{2}$$

Sampling Probability Distribution. Given a particular dataspace $\mathcal{X}$, $Pr(X')$ defines the probability of sampling dataset X'. We introduce two natural sampling schemes: *uniform* and *neighborhood* sampling. The probability distribution of sampling dataset X' in uniform and neighborhood sampling schemes are denoted using *U-Pr*(X') and *N-Pr*(X') respectively, and can be defined using the following formulas.

$$U\text{-}Pr(X') \propto \frac{1}{|\mathcal{X}|}$$

$$N\text{-}Pr(X') \propto e^{-\lambda D(X, X')} \tag{3}$$

Note that in the above formula, λ can be used to adjust the probability distribution. Using a high value for λ, assigns high probability to the datasets in the neighborhood of the original dataset. We use λ equal to 2 for our experiments. Finally, note that if we set λ to zero, then the probability distribution will be uniform. Also note that any distance function between datasets can be used in Equation (3). Here, we use the one defined in Equation (2).

Similarity Function. The similarity function measures how close the solutions for datasets X and X' are. There are two types of similarity functions: *value-based similarity* and *structure-based similarity.* The value-based similarity, denoted as $V\text{-}Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'})$, only compares the value of the objective functions for both solutions, and can be computed using the following formula:

$$V\text{-}Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'}) = \frac{F_{\text{Max}} - |F(X, \mathcal{S}^*_X) - F(X', \mathcal{S}^*_{X'})|}{F_{\text{Max}}}.$$

In the above formula, F_{Max} is the maximum possible value of the objective function which is used to normalize the similarity measure.

The structure-based similarity, compares the combinatorial structure of the obtained solutions. The definition of this similarity measure highly depends on the combinatorial structure of the solution. Fortunately, for both MaxCov and MaxInfluence problems, the optimal solution is a set of k elements, so any similarity measure among sets can be used. In this work, we compute the structural similarity between solutions $\mathcal{S}^*_X$ and $\mathcal{S}^*_{X'}$ with cardinality k as:

$$S\text{-}Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'}) = \frac{|\mathcal{S}^*_X \cap \mathcal{S}^*_{X'}|}{k}.$$

Finally, note that the above similarity measures are normalized in order to return values between zero and one.

2.3 Discussion

Notation. Given the different choices that we have for the dataspace, the sampling probability distribution, and the similarity function, we can define eight different smoothness measures. Table 1 contains the names of all these measures along with the configuration of each part of the measure.

Table 1. The eight configurations of Smoothness

	Dataspace	Sampling	Similarity
EUV-Smoothness	Exchange	Uniform	Value
EUS-Smoothness	Exchange	Uniform	Structure
ENV-Smoothness	Exchange	Neighborhood	Value
ENS-Smoothness	Exchange	Neighborhood	Structure
WUV-Smoothness	Swap	Uniform	Value
WUS-Smoothness	Swap	Uniform	Structure
WNV-Smoothness	Swap	Neighborhood	Value
WNS-Smoothness	Swap	Neighborhood	Structure

Interpretation of Smoothness. Recall that the value of Smoothness(X) represents the expected similarity of a solution sampled from $\mathcal{X}$ and the original solution $\mathcal{S}^*_X$. When smoothness is computed using uniform samples from dataspace $\mathcal{X}$, small smoothness values are an indication of the interestingness and statistical significance of the reported solution $\mathcal{S}^*_X$. In this case, small smoothness values mean that $\mathcal{S}_X^*$ cannot be obtained at random. However, when smoothness is computed from neighborhood samples, then large values of smoothness are desirable since they are indicative of the stability of $\mathcal{S}_X^*$ to small changes of the input data. Thus, in terms of smoothness, the ideal $\langle X, \mathscr{L}, F \rangle$ is one which has a small value of smoothness when sampling uniformly and a large value of smoothness when sampling in the neighborhood.

When choosing the right dataspace for smoothness evaluation of $\langle X, \mathscr{L}, F \rangle$ one must keep in mind that the exchange dataspace ($\mathcal{X}_E$) is a much larger superset of the swap dataspace ($\mathcal{X}_W$). Interestingly, depending on the nature of $\langle X, \mathscr{L}, F \rangle$ one dataspace may give more insight than the other. As an example, consider the MaxCov problem, solved by the greedy approximation algorithm. In this case, the solutions reported by the algorithm are highly dependent on the marginals of the input dataset – after all, the greedy algorithm always reports the row of the binary matrix X that has the largest number of 1s and most of the real-life datasets have heavy tailed marginal distribution. In such a case, smoothness measures computed in $\mathcal{X}_W$ would provide very limited information since samples in $\mathcal{X}_W$ maintain a constant marginal distribution and thus would tend to produce a similar solution for every sample. Therefore, the smoothness values obtained for swap dataspace would be very large. However, this large value is mainly an artifact of the algorithm used and offers limited information about the dataset itself. On the other hand, if smoothness were to be computed in $\mathcal{X}_E$ the results would be much more informative since the samples would be drawn from a much larger space allowing for more variation in the sampled datasets.

For seeing the usefulness of both the value and structural smoothness, consider the problem of identifying the right value of k for the MaxCov(k) problem. Smoothness can be used to find such k as follows: first pick a dataspace (e.g., $\mathcal{X}_E$) and compute the four smoothness measures associated with it (e.g., EUV-Smoothness(X), ENV-Smoothness(X), EUS-Smoothness(X) and ENS-Smoothness(X) for each value of k). Given that large (resp. small) values for neighborhood (resp. uniform) smoothness are desirable one can pick the k that achieves such values both in terms of the structure and the value of the solutions.

Smoothness and p-Values. There is an analogy between the smoothness score Smoothness(X) and (empirical) p-values [5,6,9,14,17,16,18,23], used to evaluate the statistical significance of the value of the solution obtained for $\langle X, \mathscr{L}, F \rangle$. However, p-values encode the probability that there exists a random dataset from $\mathcal{X}$ with solution of (almost) identical value to $\mathcal{S}_X^*$. On the other hand, Smoothness(X) is the expected similarity between solutions sampled from $\mathcal{X}$. Moreover, contrary to p-values, smoothness can be computed both with respect to the structure as well as the value of the solutions, and both for neighorhood and uniform samples.

However, the key advantage of smoothness – when compared to p-values – is the following: p-values simply *count* how many datasets have solutions with similar value as the original dataset. Smoothness takes into account the values of these solutions. In that way, smoothness provides a more comprehensive quantification of the structure of the solution space.

3 Sampling Datasets from the Dataspace

Computing Smoothness(X) precisely requires generating all possible datasets in $\mathcal{X}$ and running the optimization algorithm on all of them. This computation is infeasible since the number of datasets in both $\mathcal{X}_W$ and $\mathcal{X}_E$ is exponentially large in most cases. In this section, we explain how different smoothness measures can be accurately estimated by sampling from the exchange and the swap dataspaces.

3.1 Sampling from the Exchange Dataspace

Recall that the exchange dataspace $\mathcal{X}_E$ consists of all datasets with the same size and the same number of 1's as the input dataset X. For the rest of the discussion, we will use N_1 (resp. N_0) to denote the number 1's (reps. 0's) in X. Finally, we will use N to be the total number of entries in X: $N = N_1 + N_0$.

Uniform Sampling from the Exchange Dataspace. The following algorithm draws a sample matrix Y uniformly at random from the exchange dataspace X_E: Randomly select N_1 entries of Y. Set them to 1, and set all the other entries to 0. The random selection of N_1 entries can be done by permuting all entries of Y and picking the first N_1 entries. As a result, the overall running time of this algorithm is $O(N)$.

Neighborhood Sampling from the Exchange Dataspace. A naïve Markov Chain Monte Carlo (MCMC) algorithm for neighborhood sampling from X_E performs a random walk on the state space that consists of the distinct elements of X_E. A transition from $Y \in X_E$ to $Y' \in X_E$ happens via a single *exchange* operation. An exchange operation selects uniformly at random an 1-valued entry and a 0-valued entry from Y, and makes their values 0 and 1 respectively. Clearly, an exchange operation does not change the number of 1's in the matrix.

The number of possible exchanges for all matrices in $\mathcal{X}_E$ is the same and equal to $N_1 \times N_0$. This implies that the out-degree of each state in the mentioned Markov chain is $N_1 \times N_0$. Since exchange is a reversible operation, the in-degree of each state is also $N_1 \times N_0$. Based on these observations, we can conclude that the stationary distribution for this Markov chain is uniform.

By applying a *Metropolis-Hastings* technique [7,13] on the mentioned Markov chain, we can change the uniform distribution to the desired neighborhood distribution. With Metropolis-Hastings, we do a transition from state Y to state Y' with probability $\min\{e^{\lambda(\text{Dist}(X,Y)-\text{Dist}(X,Y'))}, 1\}$.

Metropolis-Hastings guarantees that the stationary distribution matches our definition of neighborhood sampling. Although the above MCMC-based method obtains samples from the right distribution, it is computationally expensive; The state space of the Markov chain is $\binom{N}{N_1} \approx N^{N_1}$. Also, it is not easy to prove that the Markov chain will converge in polynomially many steps.

However, we can use the following observation to design a more efficient sampling algorithm.

Observation 1. *Consider a single transition of the naïve MCMC algorithm from state Y to state Y'. Also, assume that (i,j) and (i',j') are the positions* *of the selected 0-valued and 1-valued entries for the exchange operation. This means that: $Y_{i,j} = Y'_{i',j'} = 0$ and $Y_{i',j'} = Y'_{i,j} = 1$. One can observe that based on the values in the original matrix X, $D(Y', X)$ can only take one of the following three values:*

(a) If $X_{i,j} = 1, X_{i',j'} = 0$, then $D(Y', X) = D(Y, X) - 1$.
(b) If $X_{i,j} = 0, X_{i',j'} = 1$, then $D(Y', X) = D(Y, X) + 1$.
(c) If $X_{i,j} = X_{i',j'}$, then $D(Y', X) = D(Y, X)$.

The above observation hints that in this Markov chain, not every state can transition to another state. In fact, from states W_d that are within distance d from the original matrix, we can only transition to states in W_d, W_{d-1}, and W_{d+1}. The following proposition shows that the probability of such transitions can be computed analytically.

Proposition 1. *If W_d is the set of all datasets in $\mathcal{X}_E$ such that for every $Y \in W_d$ $D(Y, X) = d$, then a single exchange operation leads to dataset Y' which belongs to W_d, or W_{d-1} or W_{d+1}. The probabilities of these events are:*

$$Pr(W_d \rightarrow W_{d-1}) = \frac{d^2}{N_1 \times N_0}, \tag{4}$$

$$Pr(W_d \rightarrow W_{d+1}) = \frac{(N_1 - d)(N_0 - d)}{N_1 \times N_0} \times e^{-\lambda}, \tag{5}$$

$$Pr(W_d \rightarrow W_d) = 1 - Pr(W_d \rightarrow W_{d+1}) - Pr(W_d \rightarrow W_{d-1}).$$

Using Proposition 1, we propose a new sampling algorithm, the *Xchange-Sampler*, which is both efficient and samples from the right (neighborhood) distribution. The pseudocode of *Xchange-Sampler* is shown in Algorithm 1.

First, the algorithm forms a Markov Chain $\mathcal{M}$ with state space $\mathcal{W}$ and a transition matrix P: $\mathcal{M} = \langle \mathcal{W}, P \rangle$. Each state W_d in $\mathcal{W}$, corresponds to datasets that are within distance d form the original dataset. The probabilities of transitions from any state W_d to states W_{d-1}, W_{d+1}, and W_d are given by the equations in Proposition 1. All these transition probabilities are summarized into the transition matrix P. Using the `StationaryProb` routine, *Xchange-Sampler* first computes the stationary probability distribution π of this Markov chain. Given π, the algorithm then samples z samples from the exchange dataspace as follows: First, using the `SampleDistance` function, it samples state W_d from $\mathcal{W}$ according to the stationary distribution π. Given W_d, the `UniformSample` function samples one of the datasets within d uniformly at random. This latter step can be implemented by randomly picking d 1-valued entries and d 0-valued entries from X, and setting them to 0 and 1 respectively.

Observe, that *Xchange-Sampler* simulates the naïve MCMC approach. The difference is that in *Xchange-Sampler*, all states that are within distance d from X are merged into a single state W_d, while in MCMC they are all distinct states. Then, the Markov chain $\mathcal{M}$ is constructed in such a way, so that in

Algorithm 1. The *Xchange-Sampler* algorithm problem.

```
Input: binary matrix $X$, integer $z$
Output: binary matrices $Y_1, Y_2, \ldots, Y_z$
$\pi \leftarrow$StationaryProb($\mathcal{W}$, $P$)
for $i = 1 \to z$ do
    $W_d \leftarrow$SampleDistance($\pi$)
    $Y_i \leftarrow$UniformSample($W_d$)
    report $Y_i$
```

the stationary probability distribution, $\pi(W_d)$ is the same as the sum of the stationary probabilities that the MCMC random walk would end in any state that is within distance d from X. Thus, *Xchange-Sampler* samples datasets by first picking their distance d (using π) and then sampling uniformly from W_d.

Observe that the number of states in the Markov chain $\mathcal{M}$ is equal to the largest distance between any two datasets in the exchange dataspace, which is at most $min\{N_1, N_0\}$. This state space is significantly smaller than the state space of the naïve MCMC method since the former is bounded by the number of entries in X, while the latter was exponential. Moreover, the naïve method performs a random walk for each sample, while *Xchange-Sampler* computes the stationary distribution π only once. Given π, *Xchange-Sampler* generates each sample in $O(N)$ time.

3.2 Sampling from the Swap Dataspace

Recall that all the datasets in the swap dataspace ($\mathcal{X}_W$) have the same row and column marginals as the original dataset X. Gionis et al. [5] have proposed an MCMC approach to obtain uniform samples from $\mathcal{X}_W$. In that MCMC method, the state space is the set of all possible matrices, and the transition from one state to another is done using an operation called *swap*. Similar to the way that exchange operations maintain the number of 1's in a binary matrix, the swap operations guarantees to maintain the row and column marginals of the matrix. To sample datasets from the neighborhood of the original dataset, we combine the method by Gionis et al. with an additional Metropolis-Hastings technique so that we sample from the correct neighborhood distribution.

4 Experiments

In this section, we present an evaluation of the different smoothness measures that we have defined. First, we demonstrate the usefulness of value smoothness in determining the statistical significance of data mining results. Next, we compare the value smoothness measures to the traditional method of statistical-significance testing using p-values. Lastly, we evaluate structural smoothness and gauge its ability to provide additional insights on the data-mining results.

4.1 Datasets

We use the following datasets in our experiments:

Cora. This dataset [19] is a binary matrix which represents a bipartite graph between terms and scientific papers[2]. The entry (i, j) in the matrix has a value of 1 if term i appears in paper j. The **Cora** matrix has size 1433×2708, density 1.26% and is characterized by a heavy tailed marginal distribution. We use **Cora** as input to the MaxCov(k) problem, where our goal is to pick a set of k terms that cover the maximum number of papers (dominant words).

Bibsonomy. This dataset [3] is a binary matrix which represents a bipartite graph between authors and scientific papers tagged with the tag "programming"[3]. The entry (i, j) in the matrix has a value of 1 if author i is a co-author of the paper j. The **Bibsonomy** matrix has 4410×3862 and density of ones: 0.049%. We use **Bibsonomy** as input to the MaxCov(k) problem, where our goal is to pick a set of k authors that cover the maximum number of publications.

Co-Authors. This dataset [12] is a collaboration network derived from the General Relativity and Quantum Cosmology category of the e-print arXiv[4]. The network contains 5242 vertices, which correspond two authors. There is an edge between two authors, if they have written a paper together. The graph contains 28968 edges and has an edge density of 0.1%. We use **Co-Authors** as input to the MaxInfluence(k) problem where our goal is to select k authors that have the maximum influence in the network.

4.2 Value Smoothness

The experiments presented in this section, aim to demonstrate the utility of value smoothness (i.e., smoothness measures where solutions are compared using value similarity). Recall that there are four different types of value smoothness (see Table 1). In order to demonstrate the utility of each type, we conducted the following experiment: first, we computed all four measures for the **Bibsonomy**, **Cora** and **Co-Authors** datasets. Figure 1 shows the corresponding values of smoothness as a function of k.

Stability of the Results. First, we observe that the neighborhood smoothness is always at least as large as uniform, in both dataspaces. This demonstrates that our optimization problems are more stable within the neighborhoods of the input datasets. For **Cora** we observe high neighborhood and low uniform smoothness making the obtained solutions stable and statistically significant. On the other hand, for **Bibsonomy** both the neighborhood and the uniform smoothness are high; this indicates that the solution obtained for this dataset is not statistically significant (i.e., could also be obtained at random). Finally, the

[2] Available at: www.cs.umd.edu/projects/linqs/projects/lbc/index.html
[3] Available at: www.kde.cs.uni-kassel.de/bibsonomy/dumps/
[4] Available at: http://snap.stanford.edu/data/ca-GrQc.html

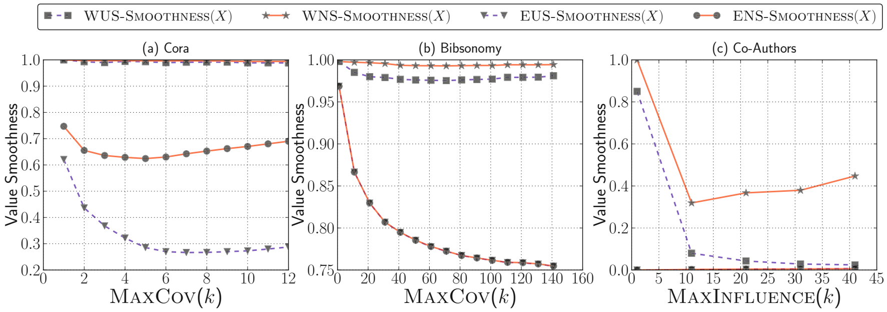


Fig. 1. Comparing the different measures of value smoothness with varying k

smoothness values for the **Co-Authors** dataset is small in all cases, indicating that the solution space is unstable even in the neighborhood of the input graph.

Advantage of the Exchange Dataspace. The second important observation one can make from Figure 1 is that smoothness of MaxCov computed using the swap dataspace is always very close to 1, independent of whether the samples were obtained using neighborhood or uniform samplimg. This is more pronounced in **Cora** , since this datasaet also has heavy-tailed marginal distribution. As we have already discussed in Section 2.3, for this dataset the smoothness computed using $\mathcal{X}_W$ does not provide as much insight as the smoothness computed using the exchange dataspace.

Advantages of the Swap Dataspace. While the exchange dataspace $\mathcal{X}_E$ can prove useful for the analysis of some datasets it might prove inadequate for the analysis of others. This is particularly true when the size of $\mathcal{X}_E$ is extremely large and any reasonable number of samples is inadequate. In such cases the swap dataspace ($\mathcal{X}_W$), which is significantly smaller, gives more insightful results. For example, in 1(c) the plots for neighborhood and uniform smoothness in $\mathcal{X}_E$ completely overlap. This overlap is an artifact of the extreme sparsity of **Co-Authors**. However, the smoothness computed using $\mathcal{X}_W$ is much more informative. It demonstrates that the dataset has high smoothness in the neighborhood making it stable and at the same time has low smoothness in the uniform distribution making it statistically significant. The same effect can also be seen in **Bibsonomy** since it is even more sparse than **Co-Authors**.

4.3 Comparing Value Smoothness with Empirical p-Values

In this section, we compare smoothness to the traditional method of statistical significance testing using empirical p-values. The results of this comparison for the exchange dataspace and uniform sampling are shown in Table 2. The table shows the EUV-Smoothness values and the empirical p-values of the solutions obtained by using **Bibsonomy** as input to the MaxCov(k) problem.

The reported p-values suggest that the solutions obtained for for the **Bibsonomy** are statistically significant. This is because the p-values for all k are equal

Table 2. Value Smoothness and p-value for the **Bibsonomy** dataset. Sampling from exchange dataspace and 10000 samples.

	Exchange Model	
MaxCov(k)	Value Smoothness	Empirical p-value
1	0.9686	0.0000
10	0.8665	0.0000
20	0.8296	0.0000
40	0.7946	0.0000
60	0.7776	0.0000
80	0.7672	0.0000
100	0.7615	0.0000

to zero, which means that the probability of sampling a dataset with best F-Cov value as large as **Bibsonomy** is negligible. On the other hand, the smoothness values are relatively high (larger than 0.7 in all cases). This suggests that *on expectation* the solutions on random datasets have very similar values to the solution obtained for **Bibsonomy**. Interestingly, these two findings seem to be contradictory.

The reason for this somewhat surprising result is the following: the p-values encode the probability that a random dataset has solution with value greater or (almost) equal to the original solution. In this case, there are no random datasets that satisfy this condition and therefore the p-values are equal to 0. On the other hand, the smoothness computes the average value of the solutions to the random datasets. Therefore, even if these solutions have indeed smaller values than the original one, they may still be relatively close to the original solution, yielding high smoothness values.

The above experiment is just one example where the value of smoothness is much more informative than the p-values. The reason for that is that the computation of smoothness takes into account the actual values of the solutions of *all* sampled datasets. The p-values on the other hand, simply count how many times the randomly sampled datasets have solutions with values equal to the original dataset. Therefore, the p-values ignore the values of these solutions and, inevitably, are less informative.

Note we obtained results similar to the above for other datasets as well. However, we ommit them, due to lack of space.

4.4 Structural Smoothness

Here, we conduct the same experiment as seen in Section 4.2, but we evaluate the structural smoothness instead of the value smoothness. The results we obtained are presented in Figure 2. Once again, we observe that the neighborhood is at least as smooth as the uniform distribution in all cases. In our problems, we can use structural smoothness to decide the optimal value of k. For example, by looking at the structural smoothness for **Cora** (Figure 2(a)), we can conclude that selecting value for $k = 5$ is not a good choice; this is because the smoothness computed over uniform samples for $k = 5$ is almost 1. This means that random datasets have identical solution to the input dataset. On the other hand, $k = 3$ or $k = 11$ are better choices since the the original solution has more differences with the random solutions. If we view the values of structural smoothness together with the value smoothness for the same dataset (Figure 1(a)), it becomes evident that $k = 11$ is the better choice for the value of k for **Cora**. This is because for $k = 11$, both the value and the strutural smoothness computed over random samples have relatively low values. Notice that the value smoothness alone would not have provided all the information necessary to choose the right value of k.

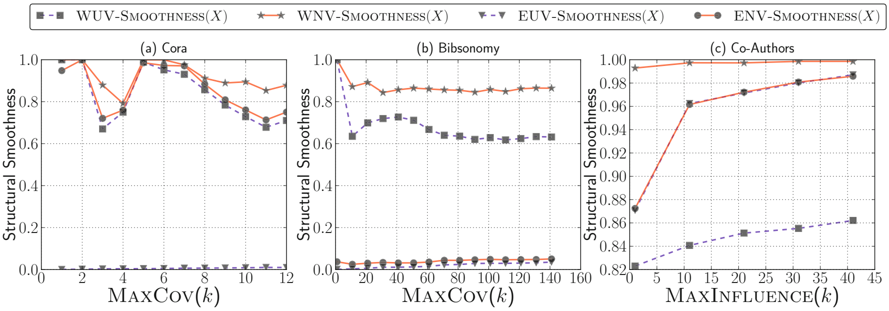


Fig. 2. Structural smoothness as a function of k

5 Related Work

To the best of our knowledge, we are the first to introduce smoothness as a measure for evaluating how small (or large) changes in the input affect the data-mining results. However, our work has connections with existing lines of research. We discuss these connections next.

Evaluation of Data-Mining Results. Existing work on data mining [5,17,16,23] and analysis of ecological and biological datasets [6,9,14,18] focuses on the evaluation of the statistical significance of the solution to $\langle X, \mathcal{S}, F\rangle$, via empirical p-values. Empirical p-values encode the probability that there exists a random dataset from $\mathcal{X}$ with solution with (almost) identical *value* to $\mathcal{S}^*_X$. On the other hand, the smoothness of a dataset is the expected similarity – in terms of value or structure – between solutions sampled from $\mathcal{X}$. However, the main difference between the smoothness framework we propose here and the empirical p-values proposed in the past is that our framework also considers non-uniform samples from $\mathcal{X}$ and explores the solutions in the "neighborhood" of X.

Moreover, our algorithms for sampling 0–1 datasets from $\mathcal{X}_E$ extend existing algorithms for uniform sampling from $\mathcal{X}_W$ [5]. However, again our algorithmic contribution goes beyond devising techniques for sampling from $\mathcal{X}_E$. For example, one of our main contributions is an efficient algorithm for sampling from the exchange dataspace $\mathcal{X}_E$.

Smoothed Analysis of Algorithms. Our work is also related to existing work on characterizing the stability of numerical methods [4,8] as well as the smoothed complexity of algorithms [1,20,21]. The key difference between our work and existing work in these areas is that we explore the smoothness of the value as well as the structure of our results. The issue of the solutions' structure has only recently appeared in the work of Balcan et al. [2]. More specifically, Balcan et al. address the problem of finding a solution that has similar structure to the optimal solution. Although related to our study, their requirement focuses on approximating the structure of the optimal solution ignoring the rest of the solution space.

Privacy-Preserving Query Answering. Our focus on the smoothness of the solutions to the input data instance connects our work with the work of Nissim et al. [15]. In that work, the authors focused on determining the amount of noise required for differentially-private query answering. Our focus is on studying how small changes in the data affect the data-mining results both in terms of value and in terms of structure.

6 Conclusions

In this paper, we presented a framework for evaluating the significance as well as the stability of of data-mining results. We achieved that by defining the notion of smoothness, which quantifies how these results get affected by small or large changes in the input data. In principle, our framework can be applied to all data-mining problems, which have been formalized using combinatorial-optimization formulations. For concreteness, we focused on a particular type of combinatorial-optimization problem, namely entity selection, and we demonstrated the application of our framework in this setting. From the computational point of view, our the evaluation of smoothness requires efficient sampling of datasets that are within a certain distance from the input dataset. As another contribution, we presented an efficient algorithm for obtaining such samples. Our preliminary experimental results demonstrates that the values of smoothness provide valuable insights about the structure of the solution space and the significance of the obtained data-mining results.

Acknowledgments. The authors want to thank John Byers for useful discussions. This research was supported by the NSF award #1017529 and by gifts from Microsoft, Yahoo! and Google.

References

1. Arthur, D., Manthey, B., Röglin, H.: k-means has polynomial smoothed complexity. In: FOCS, pp. 405–414 (2009)
2. Balcan, M.-F., Blum, A., Gupta, A.: Approximate clustering without the approximation. In: SODA, pp. 1068–1077 (2009)
3. Knowledge and Data Engineering Group, University of Kassel: Benchmark Folksonomy Data from Bibsonomy. Version of June 30 (2007)

4. Burden, R.L., Faires, J.D.: Numerical Analysis. Thomson Brooks/Cole (2005)
5. Gionis, A., Mannila, H., Mielikäinen, T., Tsaparas, P.: Assessing data mining results via swap randomization. In: KDD, pp. 167–176 (2006)
6. Haiminen, N., Mannila, H., Terzi, E.: Comparing segmentations by applying randomization techniques. BMC Bioinformatics 8 (2007)
7. Hastings, W.: Monte carlo samping methods using markov chains and their applications. Biometrika 57, 97–109 (1970)
8. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. Society of Industrial and Applied Mathematics (1996)
9. Kashtan, N., Itzkovitz, S., Milo, R., Alon, U.: Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. Bioinformatics 20(11), 1746–1758 (2004)
10. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD (2003)
11. Lappas, T., Gunopulos, D.: Efficient Confident Search in Large Review Corpora. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part II. LNCS, vol. 6322, pp. 195–210. Springer, Heidelberg (2010)
12. Leskovec, J., Kleinberg, J.M., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. TKDD 1(1) (2007)
13. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1092 (1953)
14. Milo, R., Shen-Orr, S., Itzkovirz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: Simple building blocks of complex networks. Science 298 (2002)
15. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: STOC, pp. 75–84 (2007)
16. Ojala, M., Garriga, G.C., Gionis, A., Mannila, H.: Evaluating query result significance in databases via randomizations. In: SDM, pp. 906–917 (2010)
17. Ojala, M., Vuokko, N., Kallio, A., Haiminen, N., Mannila, H.: Randomization methods for assessing data analysis results on real-valued matrices. Statistical Analysis and Data Mining 2(4), 209–230 (2009)
18. Sanderson, J.: Testing ecological patterns. American Scientist 88, 332–339 (2000)
19. Sen, P., Namata, G.M., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29(3), 93–106 (2008)
20. Spielman, D.A., Teng, S.-H.: Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In: STOC, pp. 296–305 (2001)
21. Spielman, D.A., Teng, S.-H.: Smoothed analysis: an attempt to explain the behavior of algorithms in practice. Commun. ACM 52(10), 76–84 (2009)
22. Tsaparas, P., Ntoulas, A., Terzi, E.: Selecting a comprehensive set of reviews. In: KDD, pp. 168–176 (2011)
23. Vuokko, N., Kaski, P.: Testing the significance of patterns in data with cluster structure. In: ICDM, pp. 1097–1102 (2010)

Active Evaluation of Ranking Functions Based on Graded Relevance

Christoph Sawade[1], Steffen Bickel[2], Timo von Oertzen[3], Tobias Scheffer[1], and Niels Landwehr[1]

[1] University of Potsdam, Department of Computer Science, August-Bebel-Strasse 89,
14482 Potsdam, Germany
{sawade,scheffer,landwehr}@cs.uni-potsdam.de
[2] Nokia gate5 GmbH, Invalidenstrasse 117, 10115 Berlin, Germany
steffen.bickel@nokia.com
[3] University of Virginia, Department of Psychology, Charlottesville, VA 22903
timo@virginia.edu

Abstract. Evaluating the quality of ranking functions is a core task in web search and other information retrieval domains. Because query distributions and item relevance change over time, ranking models often cannot be evaluated accurately on held-out training data. Instead, considerable effort is spent on manually labeling the relevance of query results for test queries in order to track ranking performance. We address the problem of estimating ranking performance as accurately as possible on a fixed labeling budget. Estimates are based on a set of most informative test queries selected by an active sampling distribution. Query labeling costs depend on the number of result items as well as item-specific attributes such as document length. We derive cost-optimal sampling distributions for the commonly used performance measures Discounted Cumulative Gain (DCG) and Expected Reciprocal Rank (ERR). Experiments on web search engine data illustrate significant reductions in labeling costs.

Keywords: Information Retrieval, Ranking, Active Evaluation.

1 Introduction

This paper addresses the problem of estimating the performance of a given ranking function in terms of graded relevance measures such as Discounted Cumulative Gain [1] and Expected Reciprocal Rank [2]. In informational retrieval domains, ranking models often cannot be evaluated on held-out training data. For example, older training data might not represent the distribution of queries the model is currently exposed to, or the ranking model might be procured from a third party that does not provide any training data.

In practice, ranking performance is estimated by applying a given ranking model to a representative set of test queries and manually assessing the relevance of all retrieved items for each query. We study the problem of estimating

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 676–691, 2012.

ranking performance as accurately as possible on a fixed budget for labeling item relevance, or, equivalently, minimizing labeling costs for a given level of estimation accuracy. We also study the related problem of cost-efficiently comparing the ranking performance of two models; this is required, for instance, to evaluate the result of an index update.

We assume that drawing unlabeled data $x \sim p(x)$ from the distribution of queries that the model is exposed to is inexpensive, whereas obtaining relevance labels is costly. The standard approach to estimating ranking performance is to draw a sample of test queries from $p(x)$, obtain relevance labels, and compute the empirical performance. However, recent results on *active risk estimation* [3] and *active comparison* [4] indicate that estimation accuracy can be improved by drawing test examples from an appropriately engineered instrumental distribution $q(x)$ rather than $p(x)$, and correcting for the discrepancy between p and q by importance weighting.

In this paper, we study active estimates of ranking performance. Section 2 details the problem setting. A novel aspect of active estimation in a ranking setting is that labeling costs vary according to the number of items that are relevant for a query. Section 3 derives cost-optimal sampling distributions for the estimation of DCG and ERR. Section 4 discusses empirical sampling distributions in a pool-based setting. Naïve computation of the empirical distributions is exponential, we derive polynomial-time solutions by dynamic programming. Section 5 presents empirical results. Section 6 discusses related work, Section 7 concludes.

2 Problem Setting

Let $\mathcal{X}$ denote a space of queries, and $\mathcal{Z}$ denote a finite space of items. We study ranking functions

$$\mathbf{r} : x \mapsto \left(r_1(x), \ldots, r_{|\mathbf{r}(x)|}(x)\right)^{\mathsf{T}}$$

that, given a query $x \in \mathcal{X}$, return a list of $|\mathbf{r}(x)|$ items $r_i(x) \in \mathcal{Z}$ ordered by relevance. The number of items in a ranking $\mathbf{r}(x)$ can vary depending on the query and application domain from thousands (web search) to ten or fewer (mobile applications that have to present results on a small screen). Ranking performance of $\mathbf{r}$ is defined in terms of graded relevance labels $y_z \in \mathcal{Y}$ that represent the relevance of an item $z \in \mathcal{Z}$ for the query x, where $\mathcal{Y} \subset \mathbb{R}$ is a finite space of relevance labels with minimum zero (irrelevant) and maximum y_{max} (perfectly relevant). We summarize the graded relevance of all $z \in \mathcal{Z}$ in a label vector $\mathbf{y} \in \mathcal{Y}^{\mathcal{Z}}$ with components y_z for $z \in \mathcal{Z}$.

In order to evaluate the quality of a ranking $\mathbf{r}(x)$ for a single query x, we employ two commonly used ranking performance measures: *Discounted Cumulative Gain* (DCG), given by

$$L_{dcg}\left(\mathbf{r}(x), \mathbf{y}\right) = \sum_{i=1}^{|\mathbf{r}(x)|} \ell_{dcg}\left(y_{r_i(x)}, i\right) \tag{1}$$

$$\ell_{dcg}\left(y, i\right) = \frac{2^y - 1}{\log_2(i+1)},$$

and *Expected Reciprocal Rank* (ERR), given by

$$L_{err}\left(\mathbf{r}(x),\mathbf{y}\right) = \sum_{i=1}^{|\mathbf{r}(x)|} \frac{1}{i}\ell_{err}\left(y_{r_i(x)}\right) \prod_{l=1}^{i-1}(1- \ell_{err}\left(y_{r_l(x)}\right)) \tag{2}$$

$$\ell_{err}\left(y\right) = \frac{2^y - 1}{2^{y_{max}}}$$

as introduced by Järvelin and Kekäläinen [1] and Chapelle et. al [2], respectively.

DCG scores a ranking by summing over the relevance of all documents discounted by their position in the ranking. ERR is based on a probabilistic user model: the user scans a list of documents in the order defined by $\mathbf{r}(x)$ and chooses the first document that appears sufficiently relevant; the likelihood of choosing a document z is a function of its graded relevance score y_z. If s denotes the position of the chosen document in $\mathbf{r}(x)$, then $L_{err}\left(\mathbf{r}(x),\mathbf{y}\right)$ is the expectation of the reciprocal rank $1/s$ under the probabilistic user model. Both DCG and ERR discount relevance with ranking position, ranking quality is thus most strongly influenced by documents that are ranked highly. If $\mathbf{r}(x)$ includes many items, L_{dcg} and L_{err} are in practice often approximated by only labeling items up to a certain position in the ranking or a certain relevance threshold and ignoring the contribution of lower-ranked items.

Let $p(x,\mathbf{y}) = p(x)p(\mathbf{y}|x)$ denote the joint distribution over queries $x \in \mathcal{X}$ and label vectors $\mathbf{y} \in \mathcal{Y}^{\mathcal{Z}}$ the model is exposed to. We assume that the individual relevance labels y_z for items z are drawn independently given a query x:

$$p(\mathbf{y}|x) = \prod_{z\in\mathcal{Z}} p(y_z|x,z). \tag{3}$$

This assumption is common in pointwise ranking approaches, *e.g.,* regression based ranking models [5,6]. The ranking performance of $\mathbf{r}$ with respect to $p(x,\mathbf{y})$ is given by

$$R[\mathbf{r}] = \iint L\left(\mathbf{r}(x),\mathbf{y}\right)p(x,\mathbf{y})\mathrm{d}x\,\mathrm{d}\mathbf{y}, \tag{4}$$

where $L \in \{L_{dcg}, L_{err}\}$ denotes the performance measure under study. We use integrals for notational convenience, for discrete spaces the corresponding integral is replaced by a sum. If the context is clear, we refer to $R[\mathbf{r}]$ simply by R.

Since $p(x,\mathbf{y})$ is unknown, ranking performance is typically approximated by an empirical average

$$\hat{R}_n[\mathbf{r}] = \frac{1}{n}\sum_{j=1}^{n} L\left(\mathbf{r}(x_j),\mathbf{y}_j\right), \tag{5}$$

where a set of test queries $x_1,...,x_n$ and graded relevance labels $\mathbf{y}_1,...,\mathbf{y}_n$ are drawn *iid* from $p(x,\mathbf{y})$. The empirical performance $\hat{R}_n$ consistently estimates the true ranking performance; that is, $\hat{R}_n$ converges to R with $n \to \infty$.

Test queries x_i need not necessarily be drawn according to the input distribution p. When instances are drawn according to an instrumental distribution q, a consistent estimator can be defined as

$$\hat{R}_{n,q}[\mathbf{r}] = \left(\sum_{j=1}^{n} \frac{p(x_j)}{q(x_j)}\right)^{-1} \sum_{j=1}^{n} \frac{p(x_j)}{q(x_j)} L(\mathbf{r}(x_j), \mathbf{y}_j), \tag{6}$$

where $(x_j, \mathbf{y}_j)$ are drawn from $q(x)p(\mathbf{y}|x)$ and again $L \in \{L_{dcg}, L_{err}\}$. For certain choices of the sampling distribution q, $\hat{R}_{n,q}$ may be a more label-efficient estimator of the true performance R than $\hat{R}_n$ [3].

A crucial feature of ranking domains is that labeling costs for queries $x \in \mathcal{X}$ vary with the number of items $|\mathbf{r}(x)|$ returned and item-specific features such as the length of a document whose relevance has to be determined. We denote labeling costs for a query x by $\lambda(x)$, and assume that $\lambda(x)$ is bounded away from zero by $\lambda(x) \geq \epsilon > 0$. Our goal is to minimize the deviation of $\hat{R}_{n,q}$ from R under the constraint that expected overall labeling costs stay below a budget $\Lambda \in \mathbb{R}$:

$$(q^*, n^*) = \arg\min_{q,n} \mathbb{E}\left[\left(\hat{R}_{n,q} - R\right)^2\right], \text{ s.t. } \mathbb{E}\left[\sum_{j=1}^{n} \lambda(x_j)\right] \leq \Lambda. \tag{7}$$

Note that Equation 7 represents a trade-off between labeling costs and informativeness of a test query: optimization over n implies that many inexpensive or few expensive queries could be chosen.

To estimate *relative* performance of two ranking functions $\mathbf{r}_1$ and $\mathbf{r}_2$, Equation 7 can be replaced by

$$(q^*, n^*) = \arg\min_{q,n} \mathbb{E}\left[\left(\hat{\Delta}_{n,q} - \Delta\right)^2\right], \text{ s.t. } \mathbb{E}\left[\sum_{j=1}^{n} \lambda(x_j)\right] \leq \Lambda, \tag{8}$$

where $\hat{\Delta}_{n,q} = \hat{R}_{n,q}[\mathbf{r}_1] - \hat{R}_{n,q}[\mathbf{r}_2]$ and $\Delta = R[\mathbf{r}_1] - R[\mathbf{r}_2]$. In the next section, we derive sampling distributions q^* asymptotically solving Equations 7 and 8.

3 Asymptotically Optimal Sampling

A bias-variance decomposition [7] applied to Equation 7 results in

$$\mathbb{E}\left[\left(\hat{R}_{n,q} - R\right)^2\right] = \left(\mathbb{E}\left[\hat{R}_{n,q}\right] - R\right)^2 + \mathrm{Var}\left[\hat{R}_{n,q}\right].$$

According to [8], Chapter 2.5.3, the squared bias term is of order $\frac{1}{n^2}$, while the variance is of order $\frac{1}{n}$. For large n, the expected deviation is thus dominated by the variance, and $\sigma_q^2 = \lim_{n\to\infty} n \,\mathrm{Var}[\hat{R}_{n,q}]$ exists. For large n, we can thus approximate

$$\mathbb{E}\left[\left(\hat{R}_{n,q} - R\right)^2\right] \approx \frac{1}{n}\sigma_q^2; \quad \mathbb{E}\left[\left(\hat{\Delta}_{n,q} - \Delta\right)^2\right] \approx \frac{1}{n}\tau_q^2,$$

where $\tau_q^2 = \lim_{n\to\infty} n \operatorname{Var}[\hat{\Delta}_{n,q}]$. Let $\delta(x, \mathbf{y}) = L(\mathbf{r}_1(x), \mathbf{y}) - L(\mathbf{r}_2(x), \mathbf{y})$ denote the performance difference of the two ranking models for a test query $(\mathbf{x}, y)$ and $L \in \{L_{dcg}, L_{err}\}$. The following theorem derives sampling distributions minimizing the quantities $\frac{1}{n}\sigma_q^2$ and $\frac{1}{n}\tau_q^2$, thereby approximately solving Problems 7 and 8.

Theorem 1 (Optimal Sampling for Evaluation of a Ranking Function). *Let $L \in \{L_{dcg}, L_{err}\}$ and $\sigma_q^2 = \lim_{n\to\infty} n \operatorname{Var}[\hat{R}_{n,q}]$. The optimization problem*

$$(q^*, n^*) = \arg\min_{q,n} \frac{1}{n}\sigma_q \qquad s.t. \;\; \mathbb{E}\left[\sum_{j=1}^{n} \lambda(x_j)\right] \leq \Lambda$$

is solved by

$$q^*(x) \propto \frac{p(x)}{\sqrt{\lambda(x)}} \sqrt{\int \left(L(\mathbf{r}(x), \mathbf{y}) - R\right)^2 p(\mathbf{y}|x) \mathrm{d}\mathbf{y}}, \qquad n^* = \frac{\Lambda}{\int \lambda(x) q(x) \mathrm{d}x}. \tag{9}$$

Theorem 2 (Optimal Sampling for Comparison of Ranking Functions). *Let $L \in \{L_{dcg}, L_{err}\}$ and $\tau_q^2 = \lim_{n\to\infty} n \operatorname{Var}[\hat{\Delta}_{n,q}]$. The optimization problem*

$$(q^*, n^*) = \arg\min_{q,n} \frac{1}{n}\tau_q \qquad s.t. \;\; \mathbb{E}\left[\sum_{j=1}^{n} \lambda(x_j)\right] \leq \Lambda$$

is solved by

$$q^*(x) \propto \frac{p(x)}{\sqrt{\lambda(x)}} \sqrt{\int \left(\delta(x, \mathbf{y}) - \Delta\right)^2 p(\mathbf{y}|x) \mathrm{d}\mathbf{y}}, \qquad n^* = \frac{\Lambda}{\int \lambda(x) q(x) \mathrm{d}x}. \tag{10}$$

Before we prove Theorem 1 and Theorem 2, we state the following Lemma:

Lemma 1. *Let $a : \mathcal{X} \to \mathbb{R}$ and $\lambda : \mathcal{X} \to \mathbb{R}$ denote functions on the query space such that $\int \sqrt{a(x)} \mathrm{d}x$ exists and $\lambda(x) \geq \epsilon > 0$. The functional*

$$G[q] = \left(\int \frac{a(x)}{q(x)} \mathrm{d}x\right) \left(\int \lambda(x) q(x) \mathrm{d}x\right),$$

where $q(x)$ is a distribution over the query space $\mathcal{X}$, is minimized over q by setting

$$q(x) \propto \sqrt{\frac{a(x)}{\lambda(x)}}.$$

A proof is included in the appendix. We now prove Theorem 1 and Theorem 2, building on results of Sawade et al. [3,4].

Proof (Theorem 1 and Theorem 2). We first study the minimization of $\frac{1}{n}\sigma_q^2$ in Theorem 1. Since

$$\mathbb{E}\left[\sum_{j=1}^{n} \lambda(x_j)\right] = n \int \lambda(x) q(x) \mathrm{d}x,$$

the minimization problem can be reformulated as

$$\min_q \min_n \frac{1}{n}\sigma_q^2 \text{ s.t. } n \leq \frac{\Lambda}{\int \lambda(x)q(x)\mathrm{d}x}.$$

Clearly $n^* = \Lambda / \int \lambda(x)q(x)\mathrm{d}x$ solves the inner optimization. The remaining minimization over q is

$$q^* = \arg\min_q \sigma_q^2 \int \lambda(x)q(x)\mathrm{d}x.$$

Lemma 1 in [3] implies

$$\sigma_q^2 = \iint \frac{p^2(x)}{q^2(x)} \left(L(\mathbf{r}(x),\mathbf{y}) - R\right)^2 p(\mathbf{y}|x)q(x)\mathrm{d}x\,\mathrm{d}\mathbf{y}.$$

Setting $a(x) = p^2(x)\int \left(L(\mathbf{r}(x),\mathbf{y}) - R\right)^2 p(\mathbf{y}|x)\mathrm{d}\mathbf{y}$ and applying Lemma 1 implies Equation 9. For the minimization of $\frac{1}{n}\tau_q^2$ in Theorem 2 we analogously derive

$$q^* = \arg\min_q \tau_q^2 \int \lambda(x)q(x)\mathrm{d}x.$$

Lemma 3 in [4] implies

$$\tau_q^2 = \iint \frac{p(\mathbf{x})^2}{q(\mathbf{x})^2} \left(\delta(x,\mathbf{y}) - \Delta\right)^2 p(y|\mathbf{x})q(\mathbf{x})\mathrm{d}y\,\mathrm{d}\mathbf{x}.$$

Setting $a(x) = p^2(x)\int \left(\delta(x,\mathbf{y}),\mathbf{y}) - \Delta\right)^2 p(\mathbf{y}|x)\mathrm{d}\mathbf{y}$ and applying Lemma 1 implies Equation 10. □

4 Empirical Sampling Distribution

The sampling distributions prescribed by Theorem 1 and Theorem 2 depend on the unknown test distribution $p(x)$. We now turn towards a setting in which a pool D of m unlabeled queries is available. Queries from this pool can be sampled and then labeled at a cost. Drawing queries from the pool replaces generating them under the test distribution; that is, $p(x) = \frac{1}{m}$ for all $x \in D$.

The optimal sampling distribution also depends on the true conditional $p(\mathbf{y}|x) = \prod_{z\in\mathcal{Z}} p(y_z|x,z)$ (Equation 3). To implement the method, we approximate $p(y_z|x,z)$ by a model $p(y_z|x,z;\theta)$ of graded relevance. For the large class of pointwise ranking methods – that is, methods that produce a ranking by predicting graded relevance scores for query-document pairs and then sorting documents according to their score – such a model can typically be derived from the graded relevance predictor. Finally, the sampling distributions depend on the true performance $R[\mathbf{r}]$ as given by Equation 4, or $\Delta = R[\mathbf{r}_1] - R[\mathbf{r}_2]$. $R[\mathbf{r}]$ is replaced by an introspective performance $R_\theta[\mathbf{r}]$ calculated from Equation 4, where the integral over $\mathcal{X}$ is replaced by a sum over the pool, $p(x) = \frac{1}{m}$, and $p(\mathbf{y}|x) = \prod_{z\in\mathcal{Z}} p(y_z|x,z;\theta)$. The performance difference Δ is approximated by

$\Delta_\theta = R_\theta[\mathbf{r}_1] - R_\theta[\mathbf{r}_2]$. Note that as long as $p(x) > 0$ implies $q(x) > 0$, the weighting factors ensure that such approximations do not introduce an asymptotic bias in our estimator (Equation 6).

With these approximations, we arrive at the following empirical sampling distributions.

Derivation 1. *When relevance labels for individual items are independent given the query (Equation 3), and $p(y_z|x,z)$ is approximated by a model $p(y|x,z;\theta)$ of graded relevance, the sampling distributions minimizing $\frac{1}{n}\sigma_q^2$ and $\frac{1}{n}\tau_q^2$ in a pool-based setting resolve to*

$$q^*(x) \propto \frac{1}{\sqrt{\lambda(x)}}\sqrt{\mathbb{E}\left[\left(L(\mathbf{r}(x),\mathbf{y}) - R_\theta\right)^2\Big|\, x;\theta\right]} \tag{11}$$

and

$$q^*(x) \propto \frac{1}{\sqrt{\lambda(x)}}\sqrt{\mathbb{E}\left[\left(\delta(x,\mathbf{y}) - \Delta_\theta\right)^2\Big|\, x;\theta\right]}, \tag{12}$$

respectively. Here, for any function $g(x,\mathbf{y})$ of a query x and label vector $\mathbf{y}$,

$$\mathbb{E}\left[g(x,\mathbf{y})|\, x;\theta\right] = \sum_{\mathbf{y}\in\mathcal{Y}^{\mathcal{Z}}} g(x,\mathbf{y}) \prod_{z\in\mathcal{Z}} p(y_z|x,z;\theta) \tag{13}$$

denotes expectation of $g(x,\mathbf{y})$ with respect to label vectors $\mathbf{y}$ generated according to $p(y_z|x,z,\theta)$.

We observe that for the evaluation of a single given ranking function $\mathbf{r}$ (Equation 11), the empirical sampling distribution gives preference to queries x with low costs $\lambda(x)$ and for which the expected ranking performance deviates strongly from the average expected ranking performance R_θ; the expectation is taken with respect to the available graded relevance model θ. For the comparison of two given ranking functions $\mathbf{r}_1$ and $\mathbf{r}_2$ (Equation 12), preference is given to queries x with low costs and for which the difference in performance $L(\mathbf{r}_1(x),\mathbf{y}) - L(\mathbf{r}_2(x),\mathbf{y})$ is expected to be high (note that Δ_θ will typically be small).

Computation of the empirical sampling distributions given by Equations 11 and 12 requires the computation of $\mathbb{E}\left[g(x,\mathbf{y})|\, x;\theta\right]$, which is defined in terms of a sum over exponentially many relevance label vectors $\mathbf{y} \in \mathcal{Y}^{\mathcal{Z}}$. The following theorem states that the empirical sampling distributions can nevertheless be computed in polynomial time:

Theorem 3 (Polynomial-time computation of sampling distributions).

The sampling distribution given by Equation 11 can be computed in time

$$\mathcal{O}\left(|\mathcal{Y}||D| \max_x |\mathbf{r}(x)|\right) \quad \textit{for} \;\; L \in \{L_{dcg}, L_{err}\}.$$

Algorithm 1. Active Estimation of Ranking Performance

```
input  Ranking function $\mathbf{r}$ or pair of ranking functions $\mathbf{r}_1$, $\mathbf{r}_2$; graded relevance model
    $p(y_z|x, z; \theta)$; pool $D$, labeling budget $\Lambda$.
Compute sampling distribution $q^*$ (Derivation 1, Equation 11 or 12).
Initialize $n \leftarrow 0$.
Draw $x_1 \sim q^*(x)$ from $D$ with replacement.
while $\sum_{j=1}^{n+1} \lambda(x_j) \leq \Lambda$ do
    Obtain $\mathbf{y}_{n+1} \sim p(\mathbf{y}|x_{n+1})$ from human labeler (restrict to items in rankings).
    Update number of labeled instances $n \leftarrow n + 1$.
    Draw $x_{n+1} \sim q^*(x)$ from $D$ with replacement.
end while
Compute $\hat{R}_{n,q}[\mathbf{r}]$ or $\hat{\Delta}_{n,q} = \hat{R}_{n,q}[\mathbf{r}_1] - \hat{R}_{n,q}[\mathbf{r}_2]$ (Equation 6).
output  $\hat{R}_{n,q}[\mathbf{r}]$ or $\hat{\Delta}_{n,q}$.
```

The sampling distribution given by Equation 12 can be computed in time

$$\mathcal{O}\left(|\mathcal{Y}||D| \max_x(|\mathbf{r}_1(x) \cup \mathbf{r}_2(x)|)\right) \quad \textit{for} \quad L = L_{dcg},$$

$$\mathcal{O}\left(|\mathcal{Y}||D| \max_x(|\mathbf{r}_1(x)| \cdot |\mathbf{r}_2(x)|)\right) \quad \textit{for} \quad L = L_{err}.$$

Polynomial-time solutions are derived by dynamic programming. Specifically, after substituting Equations 1 and 2 into Equations 11 and 12 and exploiting the independence assumption given by Equation 3, Equations 11 and 12 decompose into cumulative sums and products of expectations over individual item labels $y \in \mathcal{Y}$. These sums and products can be computed in polynomial time. A proof of Theorem 3 is included in the appendix.

Algorithm 1 summarizes the active estimation algorithm. It samples queries $x_1, ..., x_n$ with replacement from the pool according to the distribution prescribed by Derivation 1 and obtains relevance labels from a human labeler for all items included in $\mathbf{r}(x_i)$ or $\mathbf{r}_1(x_i) \cup \mathbf{r}_2(x_i)$ until the labeling budget Λ is exhausted. Note that queries can be drawn more than once; in the special case that the labeling process is deterministic, recurring labels can be looked up rather than be queried from the deterministic labeling oracle repeatedly. Hence, the actual labeling costs may stay below $\sum_{j=1}^{n} \lambda(x_j)$. In this case, the loop is continued until the labeling budget Λ is exhausted.

5 Empirical Studies

We compare active estimation of ranking performance (Algorithm 1, labeled *active*) to estimation based on a test sample drawn uniformly from the pool (Equation 5, labeled *passive*). Algorithm 1 requires a model $p(y_z|x, z; \theta)$ of graded relevance in order to compute the sampling distribution q^* from Derivation 1. If no such model is available, a uniform distribution $p(y_z|x, z; \theta) = \frac{1}{|\mathcal{Y}|}$ can be used instead (labeled *active*$_{uniD}$). To quantify the effect of modeling costs, we also study a variant of Algorithm 1 that assumes uniform costs $\lambda(x) = 1$

in Equations 11 and 12 (labeled $active_{uniC}$). This variant implements active risk estimation [3] and active comparison [4] for ranking; we have shown how the resulting sampling distributions can be computed in polynomial time (Derivation 1 and Theorem 3).

Experiments are performed on the Microsoft Learning to Rank data set MSLR-WEB30k [9]. It contains 31,531 queries, and a set of documents for each query whose relevance for the query has been determined by human labelers in the process of developing the Bing search engine. The resulting 3,771,125 query-document pairs are represented by 136 features widely used in the information retrieval community (such as query term statistics, page rank, and click counts). Relevance labels take values from 0 (irrelevant) to 4 (perfectly relevant).

The data are split into five folds. On one fold, we train ranking functions using different graded relevance models (details below). The remaining four folds serve as a pool of unlabeled test queries; we estimate (Section 5.1) or compare (Section 5.2) the performance of the ranking functions by drawing and labeling queries from this pool according to Algorithm 1 and the baselines discussed above. Test queries are drawn until a labeling budget Λ is exhausted. To quantify the human effort realistically, we model the labeling costs $\lambda(x)$ for a query x as proportional to a sum of costs incurred for labeling individual documents $z \in \mathbf{r}(x)$; labeling costs for a single document z are assumed to be logarithmic in the document length.

All evaluation techniques, both active and passive, can approximate L_{dcg} and L_{err} for a query x by requesting labels only for the first k documents in the ranking. The number of documents for which the MSLR-WEB30k data set provides labels varies over the queries at an average of 119 documents per query. In our experiments, we use all documents for which labels are provided for each query and for all evaluation methods under investigation.

The cost unit is chosen such that average labeling costs for a query are one. Figure 1 (left) shows the distribution of labeling costs $\lambda(x)$. All results are averaged over the five folds and 5,000 repetitions of the evaluation process. Error bars indicate the standard error.

5.1 Estimating Ranking Performance

Based on the outcome of the 2010 Yahoo ranking challenge [6,10], we choose a pointwise ranking approach and employ Random Forest regression [11] to train graded relevance models on query-document pairs. The ranking function is obtained by returning all documents associated with a query sorted according to their predicted graded relevance. We apply the approach from [12,6] to obtain the probability estimates $p(y_z|x, z; \theta)$ required by Algorithm 1 from the Random Forest model. As an alternative graded relevance model, we also study a MAP version of Ordered Logit [13]; this model directly provides probability estimates $p(y_z|x, z; \theta)$. Half of the available training fold is used for model training, the other half is used as a validation set to tune hyperparameters of the respective ranking model. Throughout the experimental evaluation, we present results for the ERR measure; results for DCG are qualitatively similar and included in [14].

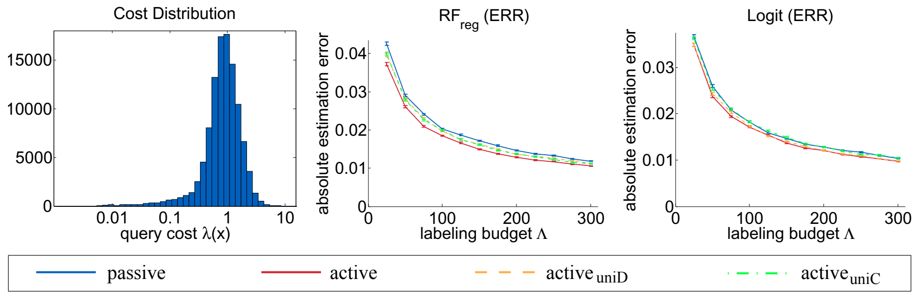


Fig. 1. Distribution of query labeling costs $\lambda(x)$ in the MSLR-WEB30k data set (left). Estimation error over Λ when evaluating Random Forest regression (center) and Ordered Logit (right). Error bars indicate the standard error.

Figure 1 (center, right) shows absolute deviation between true ranking performance and estimated ranking performance as a function of the labeling budget Λ. True performance is taken to be the performance over all test queries. We observe that active estimation is more accurate than passive estimation; the labeling budget can be reduced from $\Lambda = 300$ by about 20% (Random Forest) and 10% (Ordered Logit).

5.2 Comparing Ranking Performance

We additionally train linear Ranking SVM [15] and the ordinal classification extension to Random Forests [12,6], and compare the resulting ranking functions to those of the Ordered Logit and Random Forest regression models. For the comparison of Random Forest vs. Ordered Logit both models provide us with estimates $p(y_z|x, z; \theta)$; in this case a mixture model is employed as proposed in [4]. We measure *model selection error*, defined as the fraction of experiments in which an evaluation method does not correctly identify the model with higher true performance. Figure 2 shows model selection error as a function of the available labeling budget for different pairwise comparisons. Active estimation more reliably identifies the model with higher ranking performance, saving between 30% and 55% of labeling effort compared to passive estimation. We observe that the gains of active versus passive estimation are not only due to differences in query costs: the baseline *active$_{uniC}$*, which does not take into account query costs for computing the sampling distribution, performs almost as well as *active*.

As a further comparative evaluation we simulate an index update. An outdated index with lower coverage is simulated by randomly removing 10% of all query-document pairs from each result list $\mathbf{r}(x)$ for all queries. Random Forest regression is employed as the ranking model. Active and passive estimation methods are applied to estimate the difference in performance between models based on the outdated and current index. Figure 3 (left) shows absolute deviation of estimated from true performance difference over labeling budget Λ. We

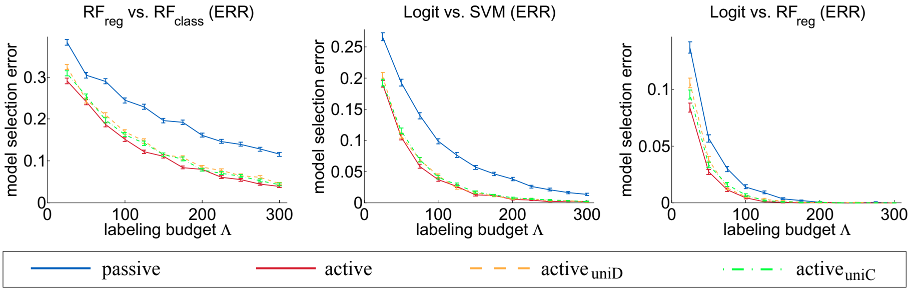


Fig. 2. Model selection error over Λ when comparing Random Forest regression vs. classification (left), and Ordered Logit vs. Ranking SVM (center) or Random Forest regression (right). Error bars indicate the standard error.

observe that active estimation quantifies the impact of the index update more accurately than passive estimation, saving approximately 75% of labeling effort.

We finally simulate the incorporation of novel sources of training data by comparing a Random Forest model trained on 100,000 query-document pairs ($\mathbf{r}_1$) to a Random Forest model trained on between 120,000 and 200,000 query-document pairs ($\mathbf{r}_2$). The difference in performance between $\mathbf{r}_1$ and $\mathbf{r}_2$ is estimated using active and passive methods. Figure 3 (center) shows absolute deviation of estimated from true performance difference for models trained on 100,000 and 200,000 instances as a function of Λ. Active estimation quantifies the performance gain from additional training data more accurately, reducing labeling costs by approximately 45%. Figure 3 (right) shows estimation error as a function of the number of query-document pairs the model $\mathbf{r}_2$ is trained on for $\Lambda = 100$. Active estimation significantly reduces estimation error compared to passive estimation for all training set sizes.

6 Related Work

There has been significant interest in learning ranking functions from data in order to improve the relevance of search results [12,16,17,6]. This has partly been driven by the recent release of large-scale datasets derived from commercial search engines, such as the Microsoft Learning to Rank datasets [9] and the Yahoo Learning to Rank Challenge datasets [10].

In this paper, we have applied ideas from active risk estimation [3] and active comparison [4] to the problem of estimating ranking performance. Our problem setting (Equations 7 and 8) generalizes the setting studied in active risk estimation and active comparison by allowing instance-specific labeling costs and constraining overall costs rather than the number of test instances that can be drawn. Applying the optimal sampling distributions derived by Sawade et al. [3,4] in a ranking setting leads to sums over exponentially many joint

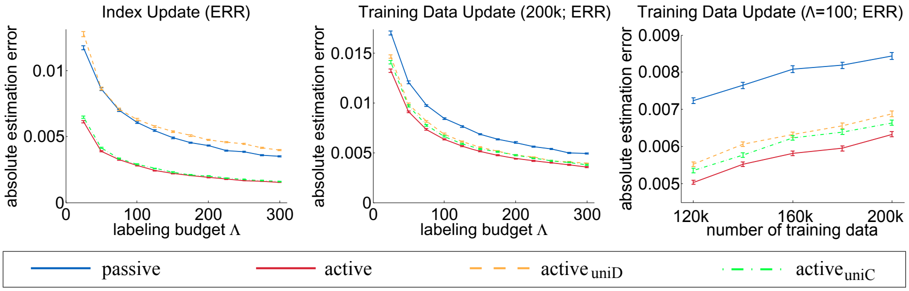


Fig. 3. Absolute estimation error over Λ for a simulated index update (left). Absolute estimation error comparing ranking functions trained on 100,000 vs. 200,000 query-document pairs over Λ (center), and over training set size of second model at $\Lambda = 100$ (right). Error bars indicate the standard error.

relevance label assignments (see Derivation 1). We have shown that they can be computed in polynomial time using dynamic programming (Theorem 3).

Besides sampling queries, it is also possible to sample subsets of documents to be labeled for a given query. Carterette et al. [18] use document sampling to decide which of two ranking functions achieves higher *precision at k*. Aslam et al. [19] use document sampling to obtain unbiased estimates of mean average precision and mean R-precision. Carterette and Smucker [20] study statistical significance testing from reduced document sets. Note that for the estimation of ERR studied in this paper, document sampling is not directly applicable because the discounting factor associated with a ranking position can only be determined if the relevance of all higher-ranked documents is known (Equation 2).

Active performance estimation can be considered a dual problem of active learning: in active learning, the goal of the selection process is to reduce the variance of predictions or model parameters; our approach reduces the variance of the performance estimate. Several active learning algorithms use importance weighting to compensate for the bias incurred by the instrumental distribution, for example in exponential family models [21] or SVMs [22].

7 Conclusions

We have studied the problem of estimating or comparing the performance of ranking functions as accurately as possible on a fixed budget for labeling item relevance. Theorems 1 and 2 derive sampling distributions that, when used to select test queries to be labeled from a given pool, asymptotically maximize the accuracy of the performance estimate. Theorem 3 shows that these optimal distributions can be computed efficiently.

Empirically, we observed that active estimates of ranking performance are more accurate than passive estimates. In different experimental settings – estimation of the performance of a single ranking model, comparison of different types of ranking models, simulated index updates – performing active estimation resulted in saved labeling efforts of between 10% and 75%.

Appendix

Proof of Lemma 1

We have to minimize the functional

$$\left(\int \frac{a(x)}{q(x)}\mathrm{d}x\right)\left(\int \lambda(x)q(x)\mathrm{d}x\right) \tag{14}$$

in terms of q under the constraints $\int q(x)\mathrm{d}x = 1$ and $q(x) > 0$. We first note that Objective 14 is invariant under multiplicative rescaling of $q(x)$, thus the constraint $\int q(x)\mathrm{d}x = 1$ can be dropped during optimization and enforced in the end by normalizing the unconstrained solution. We reformulate the problem as

$$\min_q C\int \frac{a(x)}{q(x)}\mathrm{d}x \quad \text{s.t.} \quad C = \int \lambda(x)q(x)\mathrm{d}x \tag{15}$$

which we solve using a Lagrange multiplier α by

$$\min_q C\int \frac{a(x)}{q(x)}\mathrm{d}x + \alpha\left(\int \lambda(x)q(x)\mathrm{d}x - C\right).$$

The optimal point for the constrained problem satisfies the Euler-Lagrange equation

$$\alpha\lambda(x) = C\frac{a(x)}{q(x)^2},$$

and therefore

$$q(x) = \sqrt{C\frac{a(x)}{\alpha\lambda(x)}}. \tag{16}$$

Resubstitution of Equation 16 into the constraint (Equation 15) leads to

$$C = \int \sqrt{C\frac{a(x)}{\alpha\lambda(x)}}\lambda(x)\mathrm{d}x, \tag{17}$$

solving for α we obtain

$$\alpha = \frac{\left(\int \sqrt{Ca(x)\lambda(x)}\mathrm{d}x\right)^2}{C^2}. \tag{18}$$

Finally, resubstitution of Equation 18 into Equation 16 proves the claim. □

Proof of Theorem 3

In order to show that the empirical sampling distributions given by Equations 11 and 12 can be computed efficiently, we have to show that Equation 13 can be computed efficiently. This can be done by suitable algebraic manipulation, exploiting the independence assumption given by Equation 3.

We now present the proof for the empirical distribution for absolute estimation (Equation 11) with $L = L_{err}$. The remaining cases can be found in [14]. It suffices to show that the intrinsic risk R_θ can be computed in time $\mathcal{O}(|\mathcal{Y}||D| \max_x |\mathbf{r}(x)|)$, and that for any $x \in \mathcal{X}$ the quantity $\mathbb{E}[(L(\mathbf{r}(x), \mathbf{y}) - R_\theta)^2 | x, \theta]$ can be computed in time $\mathcal{O}(|\mathcal{Y}||\mathbf{r}(x)|)$ given R_θ. We first note that for any $z \in \mathcal{Z}$, it holds that

$$\begin{aligned}
\mathbb{E}\left[\ell_{err}\left(y_z\right) \middle| x; \theta\right] &= \sum_{\mathbf{y} \in \mathcal{Y}^{\mathcal{Z}}} \ell_{err}\left(y_z\right) \prod_{z' \in \mathcal{Z}} p(y_{z'}|x, z'; \theta) \\
&= \sum_{y_z} \sum_{\mathbf{y} \in \mathcal{Y}^{\mathcal{Z} \setminus \{z\}}} \ell_{err}\left(y_z\right) \prod_{z' \in \mathcal{Z}} p(y_{z'}|x, z'; \theta) \\
&= \sum_{y_z} \ell_{err}\left(y_z\right) p(y_z|x, z; \theta) \sum_{\mathbf{y} \in \mathcal{Y}^{\mathcal{Z} \setminus \{z\}}} \prod_{z' \in \mathcal{Z} \setminus \{z\}} p(y_{z'}|x, z'; \theta) \\
&= \sum_{y_z} \ell_{err}\left(y_z\right) p(y_z|x, z; \theta), \qquad (19)
\end{aligned}$$

where $\mathbf{y} \in \mathcal{Y}^{\mathcal{Z} \setminus \{z\}}$ is a vector of relevance labels $y_{z'}$ for all $z' \in \mathcal{Z} \setminus \{z\}$. The quantity $\mathbb{E}\left[\ell_{err}\left(y_z, i\right) \middle| x; \theta\right]$ can thus be computed in time $\mathcal{O}(|\mathcal{Y}|)$. Furthermore, for $L = L_{err}$ it holds that

$$\begin{aligned}
R_\theta &= \sum_{x \in D} \frac{1}{|D|} \sum_{\mathbf{y} \in \mathcal{Y}^{\mathcal{Z}}} \sum_{i=1}^{|\mathbf{r}(x)|} \frac{1}{i} \ell_{err}\left(y_{r_i(x)}\right) \prod_{l=1}^{i-1} (1 - \ell_{err}\left(y_{r_l(x)}\right)) \prod_{z \in \mathcal{Z}} p(y_z|x, z; \theta) \\
&= \sum_{x \in D} \frac{1}{|D|} \sum_{i=1}^{|\mathbf{r}(x)|} \frac{1}{i} \mathbb{E}\left[\ell_{err}\left(y_{r_i(x)}\right) \prod_{l=1}^{i-1} (1 - \ell_{err}\left(y_{r_l(x)}\right)) \middle| x; \theta\right] \\
&= \sum_{x \in D} \frac{1}{|D|} \sum_{i=1}^{|\mathbf{r}(x)|} \frac{1}{i} \mathbb{E}\left[\ell_{err}\left(y_{r_i(x)}\right) \middle| x; \theta\right] \prod_{l=1}^{i-1} \left(1 - \mathbb{E}\left[\ell_{err}\left(y_{r_l(x)}\right) \middle| x; \theta\right]\right). \qquad (20)
\end{aligned}$$

Equation 20 can now be computed in time $\mathcal{O}(|\mathcal{Y}||D| \max_x |\mathbf{r}(x)|)$: for a given $x \in D$, we can compute the cumulative products $\prod_{l=1}^{i-1} \left(1 - \mathbb{E}\left[\ell_{err}\left(y_{r_l(x)}\right) \middle| x; \theta\right]\right)$ for $i = 1, ..., |\mathbf{r}(x)|$ in time $\mathcal{O}(|\mathcal{Y}||\mathbf{r}(x)|)$. We start by precomputing the cumulative products for all $x \in D$ and $i = 1, ..., |\mathbf{r}(x)|$ in time $\mathcal{O}(|\mathcal{Y}||D| \max_x |\mathbf{r}(x)|)$. Given precomputed cumulative products, the final summation over $x \in D$ and i can be carried out in time $\mathcal{O}(|D| \max_x |\mathbf{r}(x)|)$. We now turn towards the quantity

$$\mathbb{E}[(L(\mathbf{r}(x), \mathbf{y}) - R_\theta)^2 | x, \theta].$$

Let $\bar{\ell}_i = \frac{1}{i} \ell_{err}\left(y_i\right) \prod_{k=1}^{i-1} (1 - \ell_{err}\left(y_l\right))$. We derive

$$\begin{aligned}
&\mathbb{E}\left[\left(L\left(\mathbf{r}(x), \mathbf{y}\right) - R_\theta\right)^2 \middle| x;\theta\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{|\mathbf{r}(x)|} \bar{\ell}_i^2 + 2 \sum_{i=1}^{|\mathbf{r}(x)|} \sum_{l=i+1}^{|\mathbf{r}(x)|} \bar{\ell}_i \bar{\ell}_l - 2R_\theta \sum_{i=1}^{|\mathbf{r}(x)|} \bar{\ell}_i + R_\theta^2 \right] &\quad (21)\\
&= \sum_{i=1}^{|\mathbf{r}(x)|} \left(\mathbb{E}\left[\bar{\ell}_i^2 \middle| x;\theta\right] + 2 \sum_{l=i+1}^{|\mathbf{r}(x)|} \mathbb{E}\left[\bar{\ell}_i \bar{\ell}_l \middle| x;\theta\right] - 2R_\theta \mathbb{E}\left[\bar{\ell}_i \middle| x;\theta\right] \right) + R_\theta^2. &\quad (22)
\end{aligned}$$

We expand the square of sums twice in Equation 21. Equation 22 follows from the independence assumption (Equation 3). We note that for $l > i$ the following decomposition holds:

$$\bar{\ell}_l = \frac{1}{l} \ell_{err}\left(y_l\right) \left(\prod_{k=1}^{i-1} \left(1 - \ell_{err}\left(y_k\right)\right) \right) \left(1 - \ell_{err}\left(y_i\right)\right) \left(\prod_{k=i+1}^{l-1} \left(1 - \ell_{err}\left(y_k\right)\right) \right).$$

Thus, Equation 22 can be expressed as

$$\begin{aligned}
&\sum_{i=1}^{|\mathbf{r}(x)|} \left(\frac{1}{i^2} \mathbb{E}\left[\ell_{err}\left(y_i\right)^2 \middle| x;\theta\right] \prod_{l=1}^{i-1} \mathbb{E}\left[\left(1 - \ell_{err}\left(y_l\right)\right)^2 \middle| x;\theta\right] \right.\\
&\quad + 2\frac{1}{i} \mathbb{E}\left[\ell_{err}\left(y_i\right)\left(1 - \ell_{err}\left(y_i\right)\right) \middle| x;\theta\right] \left(\prod_{l=1}^{i-1} \mathbb{E}\left[\left(1 - \ell_{err}\left(y_l\right)\right)^2 \middle| x;\theta\right] \right)\\
&\quad\quad \cdot \left(\prod_{l=i+1}^{|\mathbf{r}(x)|} \mathbb{E}\left[\left(1 - \ell_{err}\left(y_l\right)\right) \middle| x;\theta\right] \right) \sum_{k=i+1}^{|\mathbf{r}(x)|} \frac{\mathbb{E}\left[\ell_{err}\left(y_k\right) \middle| x;\theta\right]}{k \prod_{l=k}^{|\mathbf{r}(x)|} \mathbb{E}\left[\left(1 - \ell_{err}\left(y_l\right)\right) \middle| x;\theta\right]}\\
&\quad \left. - 2R_\theta \frac{1}{i} \mathbb{E}\left[\ell_{err}\left(y_i\right) \middle| x;\theta\right] \prod_{l=1}^{i-1} \mathbb{E}\left[\left(1 - \ell_{err}\left(y_l\right)\right) \middle| x;\theta\right] \right) + R_\theta^2 \qquad (23)
\end{aligned}$$

Equation 23 can be evaluated in time $\mathcal{O}(|\mathcal{Y}||\mathbf{r}(x)|)$ as follows. We start by precomputing all cumulative products in time $\mathcal{O}(|\mathcal{Y}||\mathbf{r}(x)|)$ as shown above. The cumulative sums of the form $\sum_{k=i+1}^{|\mathbf{r}(x)|} \ldots$ for $i = |\mathbf{r}(x)| - 1, \ldots, 1$ can then be computed in overall time $\mathcal{O}(|\mathcal{Y}||\mathbf{r}(x)|)$. Given these precomputed quantities, the outer summation can then be carried out in time $\mathcal{O}(|\mathcal{Y}||\mathbf{r}(x)|)$ as well.

References

1. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems 20(4), 422–446 (2002)
2. Chapelle, O., Metzler, D., Zhang, Y., Grinspan, P.: Expected reciprocal rank for graded relevance. In: Proceeding of the Conference on Information and Knowledge Management (2009)

3. Sawade, C., Bickel, S., Scheffer, T.: Active risk estimation. In: Proceedings of the 27th International Conference on Machine Learning (2010)
4. Sawade, C., Landwehr, N., Scheffer, T.: Active comparison of prediction models (unpublished manuscript)
5. Cossock, D., Zhang, T.: Statistical analysis of Bayes optimal subset ranking. IEEE Transactions on Information Theory 54(11), 5140–5154 (2008)
6. Mohan, A., Chen, Z., Weinberger, K.: Web-search ranking with initialized gradient boosted regression trees. In: JMLR: Workshop and Conference Proceedings, vol. 14, pp. 77–89 (2011)
7. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation 4, 1–58 (1992)
8. Liu, J.S.: Monte carlo strategies in scientific computing. Springer (2001)
9. Microsoft Research: Microsoft learning to rank datasets, `http://research.microsoft.com/en-us/projects/mslr/` (released June 16, 2010)
10. Chapelle, O., Chang, Y.: Yahoo! Learning to rank challenge overview. In: JMLR: Workshop and Conference Proceedings, vol. 14, pp. 1–24 (2011)
11. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
12. Li, P., Burges, C., Wu, Q.: Learning to rank using classification and gradient boosting. In: Advances in Neural Information Processing Systems (2007)
13. McCullagh, P.: Regression models for ordinal data. Journal of the Royal Statistical Society. Series B (Methodological) 42(2), 109–142 (1980)
14. Sawade, C., Bickel, S., von Oertzen, T., Scheffer, T., Landwehr, N.: Active evaluation of ranking functions based on graded relevance. Technical report, University of Potsdam (2012)
15. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. Advances in Large Margin Classifiers, 115–132 (2000)
16. Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., Sun, G.: A general boosting method and its application to learning ranking functions for web search. In: Advances in Neural Information Processing Systems (2007)
17. Burges, C.: RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research (2010)
18. Carterette, B., Allan, J., Sitaraman, R.: Minimal test collections for retrieval evaluation. In: Proceedings of the 29th SIGIR Conference on Research and Development in Information Retrieval (2006)
19. Aslam, J., Pavlu, V., Yilmaz, E.: A statistical method for system evaluation using incomplete judgments. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval (2006)
20. Carterette, B., Smucker, M.: Hypothesis testing with incomplete relevance judgments. In: Proceedings of the 16th ACM Conference on Information and Knowledge Management (2007)
21. Bach, F.: Active learning for misspecified generalized linear models. In: Advances in Neural Information Processing Systems (2007)
22. Beygelzimer, A., Dasgupta, S., Langford, J.: Importance weighted active learning. In: Proceedings of the International Conference on Machine Learning (2009)

Community Trend Outlier Detection Using Soft Temporal Pattern Mining

Manish Gupta[1], Jing Gao[2], Yizhou Sun[1], and Jiawei Han[1]

[1] UIUC, IL
{gupta58,sun22,hanj}@illinois.edu
[2] SUNY, Buffalo, NY
jing@buffalo.edu

Abstract. Numerous applications, such as bank transactions, road traffic, and news feeds, generate temporal datasets, in which data evolves continuously. To understand the temporal behavior and characteristics of the dataset and its elements, we need effective tools that can capture evolution of the objects. In this paper, we propose a novel and important problem in evolution behavior discovery. Given a series of snapshots of a temporal dataset, each of which consists of evolving communities, our goal is to find objects which evolve in a dramatically different way compared with the other community members. We define such objects as *community trend outliers*. It is a challenging problem as evolutionary patterns are hidden deeply in noisy evolving datasets and thus it is difficult to distinguish anomalous objects from normal ones. We propose an effective two-step procedure to detect community trend outliers. We first model the normal evolutionary behavior of communities across time using soft patterns discovered from the dataset. In the second step, we propose effective measures to evaluate chances of an object deviating from the normal evolutionary patterns. Experimental results on both synthetic and real datasets show that the proposed approach is highly effective in discovering interesting community trend outliers.

1 Introduction

A large number of applications generate temporal datasets. For example, in our everyday life, various kinds of records like credit, personnel, financial, judicial, medical, etc. are all temporal. Given a series of snapshots of a temporal dataset, analysts often perform community detection for every snapshot with the goal of determining the intrinsic grouping of objects in an unsupervised manner. By analyzing a series of snapshots, we can observe that these communities evolve in a variety of ways – communities contract, expand, merge, split, appear, vanish, or re-appear after a time period. Most of the objects within a community follow similar evolution trends which define the evolution trends of the community. However, evolution behavior of certain objects is quite different from that of their respective communities. Our goal is to detect such anomalous objects as *Community Trend Outliers* (or *CTOutliers*) given community distributions of

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 692–708, 2012.

each object for a series of snapshots. In the following, we present *CTOutlier* examples and discuss importance of identifying such outliers in real applications.

***CTOutlier* Examples**

Consider the co-authorship network for the four areas in CS: data mining (DM), information retrieval (IR), databases (DB) and machine learning (ML). Every author can be associated with a soft distribution of their belongingness to each of these areas. One such sequence of distributions could be $\langle$*1:(DB:1*, *DM:0)*, *2:(DB:0.8*, *DM:0.2)*, *3:(DB:0.5*, *DM:0.5)*, *4:(DB:0.1*, *DM:0.9)*$\rangle$. Such a pattern represents the trend of a part of DB researchers gradually moving into the DM community. While most of the authors follow one of such popular patterns of evolution with respect to their belongingness distributions across different snapshots, evolution of the distributions associated with some of the other authors is very different. Such authors can be considered as *CTOutliers*.

As another example, consider all the employees working for a company. For each employee, one can record the amount of time spent in Office work, Household work, Watching TV, Recreation and Eating, for a month. Across different days, one can observe a trend where a person spends most of his time in office work on weekdays and in household work on weekends. Similarly, there could be different patterns for night workers. However, there could be a very few employees who follow different schedule for a few days (e.g., if an employee is sick, he might spend a lot of his time at home even on weekdays). In that case, that employee can be considered as a *CTOutlier*.

Besides these examples, interesting examples of *CTOutliers* can be commonly observed in real-life scenarios. A city with a very different sequence of land use proportion (agriculture, residential, commercial, open space) changes across time, compared to change patterns for other cities can be a *CTOutlier*. E.g., most of the cities show an increase in residential and commercial areas and reduction in agriculture areas over time. However, some cities may get devastated by natural calamities disturbing the land use drastically. Applications where *CTOutliers* could be useful depends on the specific domain. Outlier detection may be useful to explain future behavior of outlier sequences. E.g., one may analyze the diet proportion of carbohydrates, proteins and fats for a city across time. A city showing trends of increasing fats proportion in diet, may have a population with larger risk of heart attacks. *CTOutlier* detection may be used to trigger action in monitoring systems. E.g., in a chemical process, one may expect to observe a certain series of distribution of elements across time. Unexpected deviations from such a series, may be used to trigger a corrective action.

Brief Overview of *CTOutlier* Detection

We study the problem of detecting *CTOutliers* given community distributions of each object for a series of snapshots of a temporal dataset. Input for our problem thus consists of a soft sequence (i.e., a sequence of community distributions across different timestamps) associated with each object. For example, in DBLP, an author has a sequence of research-area distributions across years. The number of communities could change over time, so a soft sequence can consist of distributions of different sizes at different timestamps.

This problem is quite different from trajectory outlier detection [8,14,19] because: (1) In this problem, soft sequences consist of distributions obtained by community detection rather than locations in trajectory outlier detection. (2) Soft patterns could be gapped and multi-level while trajectories are usually continuous. (3) Unlike trajectory based systems, we cannot rely on additional features such as speed or direction of motion of objects. Moreover, existing efforts on detecting outliers in evolving datasets [4,15] cannot detect temporal community trend based outliers because they do not involve any notion of communities. In the first step of discovering normal trends, probabilistic sequential pattern mining methods [9,21] can be used to extract temporal patterns, however the patterns detected by these methods are "hard patterns" which are incapable of capturing subtle trends, as discussed in Sec. 2.

We propose to tackle this problem using a two-step approach: pattern extraction and outlier detection. In the pattern extraction phase, we first perform clustering of soft sequences for individual snapshots. The cluster centroids obtained for each timestamp represent the length-1 frequent soft patterns for that timestamp. Support of a length-1 pattern (cluster centroid) is defined as the sum of a function of the distance between a point (sequence) and cluster centroid, over all points. The Apriori property [5] is then exploited to obtain frequent soft patterns of length ≥ 2. After obtaining the frequent soft patterns, outlier detection is performed. A sequence is considered a *CTOutlier* if it deviates a lot from its best matching pattern for many combinations of timestamps.

In summary, we make the following contributions in this paper.

- We introduce the notion of Community Trend Outliers *CTOutliers*. Such a definition tightly integrates the notion of deviations with respect to both the temporal and community dimensions.
- The proposed integrated framework consists of two novel stages: efficient discovery of a novel kind of patterns called *soft patterns*, and analysis of such patterns using a new outlier detection algorithm.
- We show interesting and meaningful outliers detected from multiple real and synthetic datasets.

Our paper is organized as follows. *CTOutliers* are defined as objects that deviate significantly from a novel kind of patterns. Thus, pattern discovery is the basis of the outlier detection step. Hence, first we introduce the notion of *soft patterns* and develop our method to extract temporal community trends in the form of frequent soft patterns in Sec. 2. Then, in Sec. 3, we discuss the algorithm for *CTOutlier* detection which exploits the extracted patterns to compute outlier scores. We discuss the datasets and results with detailed insights in Sec. 4. Finally, related work and conclusions are presented in Sec. 5 and 6 respectively.

2 Temporal Trends Extraction

In this section, we discuss how to extract soft patterns as temporal trends, which serve as the basis of community trend outlier detection in Sec. 3. We first introduce important definitions in Sec. 2.1. Next, we will carefully define support for such soft patterns and discuss how to extract them from the soft sequence data in Sec. 2.2 and Sec. 2.3.

2.1 Problem Formulation

Let us begin with a few definitions. We will use the toy dataset shown in Fig. 1 as a running example. The toy dataset has 15 objects which consist of 4 patterns (▲, ◀, ▼, ▶) and two outliers (■, ★) across 3 timestamps. There are 3 (A, B, C), 3 (D, E, F) and 4 (G, H, I, J) clusters for the 3 timestamps respectively.

Soft Sequence: A soft sequence for object o is denoted by $S_o = \langle S_{1_o}, S_{2_o}, \ldots, S_{T_o} \rangle$ where S_{t_o} denotes the community belongingness probability distribution for object o at time t. In Fig. 1, for the point marked with a → across all 3 timestamps, the soft sequence is $\langle$*1: (A:0.1, B:0.8, C:0.1), 2: (D:0.07, E:0.08, F:0.85), 3: (G:0.08, H:0.8, I:0.08, J:0.04)*$\rangle$. Soft sequences for all objects are defined on the same set of T timestamps; all sequences are synchronized in time. For a particular timestamp, the data can be represented as $S_t = [S_{t_1}, S_{t_2}, \ldots, S_{t_N}]^T$.

Soft Pattern: A *soft pattern* $p = \langle P_{1_p}, P_{2_p}, \ldots, P_{T_p} \rangle$ is a sequence of probability distributions across L_p (possibly gapped) out of T timestamps, with support ≥ *min_sup*. Here, P_{t_p} denotes the community (probability) distribution at timestamp t. A soft pattern p defined over a set τ_p of timestamps is a representative of a set of sequences (similar to each other for timestamps $\in \tau_p$) grouped together by clustering over individual snapshots. Support of p is naturally defined proportional to the number of sequences it represents (Sec. 2.2 and 2.3). E.g., the pattern $p = \langle$*1:(DB:1, DM:0), 2:(DB:0.5, XML:0.3, DM:0.2), 4:(DB:0.1, DM:0.9)*$\rangle$ is defined over 3 timestamps 1, 2 and 4 and so L_p=3. In Fig. 1, $\langle$*1:(A:0.2, B:0.2, C:0.6), 2:(D:0.9, E:0.05, F:0.05), 3:(G:0.9, H:0.03, I:0.03, J:0.04)*$\rangle$ is a soft pattern covering the ▶ points.

CTOutlier: An object o is a *CTOutlier* if its outlier score ranks within the top-k. The outlier score of an object o captures the degree to which it deviates from the best matching soft pattern across different combinations of timestamps (details in Sec. 3). E.g., outliers ★ and ■ deviate from the ▶ and ◀ patterns for the first and the third timestamps respectively.

The *CTOutlier* detection problem can then be specified as follows.

Input: Soft sequences (each of length T) for N objects, denoted by matrix S.

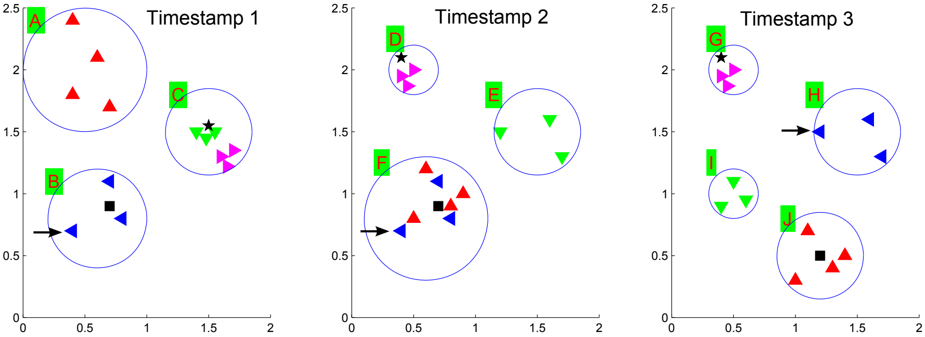


Fig. 1. Three Snapshots of a Toy Dataset

Output: Set of *CTOutlier* objects.

Before presenting the soft pattern discovery problem and the methodology for their efficient discovery, we discuss the reasons why we use soft rather than hard patterns to represent temporal trends.

Why use Soft Patterns?

Given soft sequence data, one can use probabilistic sequential pattern mining [9,21] to discover hard sequential patterns. In the DBLP example, a hard pattern can be $\langle 1{:}DB,\ 2{:}DB,\ 3{:}DM,\ 4{:}DM\rangle$, which expresses major transitions for an author changing his research area from DB to DM. However, most trends in temporal datasets are subtle and thus cannot be expressed using hard patterns. E.g., in Fig. 1, evolution of ▶ objects can be characterized by hard pattern $\langle 1{:}C,\ 2{:}D,\ 3{:}G\rangle$. However, at the first timestamp, ▶ objects lie on the boundary of cluster C and are much different from the core cluster-C ▼ objects. Such subtle difference cannot be encoded in hard patterns.

Different from hard patterns, soft patterns contain a richer set of information. Consider two soft patterns which are not related to such drastic changes: $p_1 = \langle 1{:}(DM{:}0.8,\ IR{:}0.2),\ 2{:}(DM{:}0.6,\ IR{:}0.4)\rangle$ and $p_2 = \langle 1{:}(DM{:}0.48,\ DB{:}0.42,\ IR{:}0.1),\ 2{:}(DM{:}1)\rangle$ both of which map to the same hard pattern $\langle 1{:}DM,\ 2{:}DM\rangle$. This is not reasonable because clearly the two patterns represent two different semantic trends. On the other hand, let $\langle 1{:}DB,\ 2{:}DM\rangle$ denote a hard pattern, from which we cannot identify how the communities indeed evolve. The temporal trend could be that $\langle 1{:}(DB{:}0.5,\ Sys{:}0.3,\ Arch{:}0.2),\ 2{:}(DM{:}0.5,\ DB{:}0.3,\ Sys{:}0.2)\rangle$ is frequent but $\langle 1{:}(DB{:}0.9,\ Sys{:}0.1),\ 2{:}(DM{:}0.9,\ DB{:}0.1)\rangle$ is not frequently observed. Therefore, soft patterns are more desirable because they can express that core-DB authors did not move to DM, although some non-core-DB researchers started showing interest in DM. In Fig. 1, hard pattern based detection will miss ★ outlier, while soft pattern based detection will correctly identify it. To prevent such loss of information when using hard patterns, we propose to model temporal trends as soft patterns.

Soft Pattern Discovery Problem

The soft pattern discovery problem can be summarized as follows.

Input: Soft sequences (each of length T) for N objects, denoted by matrix S.

Output: Set P of frequent soft patterns with support $\geq min\_sup$.

Next, we will carefully define support for such soft patterns and discuss how to extract them from the soft sequence data. We will first discuss how to find length-1 patterns, and then discuss how to find patterns with length ≥ 2.

2.2 Extraction of Length-1 Soft Patterns

The task of discovering length-1 soft patterns for a particular timestamp is to find representative distributions from a space of N probability distributions where N = #objects. We solve this problem by performing *clustering* (we use *XMeans* [22]) on distributions. *The cluster centroids for such clusters are a representative of all the points within the cluster. Thus, a cluster centroid can be used to uniquely represent a length-1 soft pattern.* In the example shown in Fig. 1, for the first timestamp, *XMeans* discovers 4 clusters with centroids $(A:0.85, B:0.05, C:0.1)$, $(A:0.03, B:0.9, C:0.07)$, $(A:0.03, B:0.02, C:0.95)$ and $(A:0.2, B:0.2, C:0.6)$. Each of these cluster centroids represents a length-1 soft pattern (▲, ◀, ▼, ▶ resp).

Defining Support for Length-1 Soft Patterns

Traditionally, support for a sequential pattern is defined as the number of sequences which contain that pattern. Similarly, we can define support for a soft pattern (cluster centroid) P_{t_p} in terms of the degree to which the sequences (points) belong to the corresponding cluster (Eq. 1). Let $Dist(P_{t_p}, S_{t_o})$ be some distance measure (we use Euclidean distance) between the sequence distribution for object o at time t and the cluster centroid for pattern p at time t. Let $maxDist(P_{t_p})$ be the maximum distance of any point in the dataset from the centroid P_{t_p}. Then the support for the length-1 pattern p can be expressed as follows.

$$sup(P_{t_p}) = \sum_{o=1}^{N} \left[1 - \frac{Dist(P_{t_p}, S_{t_o})}{maxDist(P_{t_p})} \right] \tag{1}$$

From Eq. 1, one can see that an object which is closer to the cluster centroid contributes more to the support of a pattern (corresponding to that cluster centroid) compared to objects far away from the cluster centroid. E.g., at the first timestamp, cluster centroid $(A:0.85, B:0.05, C:0.1)$ gets good support from all the ▲ points because they are very close to it, but gets small amount of support from other points, based on their distance from it. Patterns with support $\geq min\_sup$ are included in the set of frequent patterns P.

A clustering algorithm may break a semantic cluster into multiple sub-clusters. Hence, some of the resulting clusters may be very small and so if we define support for a cluster centroid based on just the points within the cluster, we might lose some important patterns for lack of support. Hence, we define support for a cluster centroid using contributions from all the points in the dataset. To prevent the impact of distance based outliers (when computing $maxDist$), it might be beneficial to remove such outliers from each snapshot, as a preprocessing step.

2.3 Extraction of Longer Soft Patterns

Here we will discuss how to define support for longer patterns and compute them efficiently.

Defining Support for Longer Soft Patterns

The support by an object o for a pattern p is defined in terms of its support with respect to each of the timestamps in τ_p as shown below.

$$sup(p) = \sum_{o=1}^{N} \prod_{t \in \tau_p} \left[1 - \frac{Dist(P_{t_p}, S_{t_o})}{maxDist(P_{t_p})} \right] \tag{2}$$

Intuitively, an object for which the community distribution lies close to the cluster centroids of the pattern across a lot of timestamps will have higher support contribution for the pattern compared to objects which lie far away from the pattern's cluster centroids. As an example, consider the pattern ⟨*1:(A:0.85*, *B:0.05*, *C:0.1)*, *2:(D:0.1*, *E:0.2*, *F:0.7)*, *3:(G:0.01*, *H:0.02*, *I:0.03*, *J:0.94)*⟩. The ▲ points contribute maximum support to this pattern because they lie very close to this pattern across all timestamps. Patterns with support $\geq min\_sup$ are included in the set P of frequent patterns.

Apriori Property

From Eqs. 1 and 2, it is easy to see that a soft pattern cannot be frequent unless all its sub-patterns are also frequent. Thus, the Apriori property [5] holds. This means that longer frequent soft patterns can be discovered by considering only those candidate patterns which are obtained from shorter frequent patterns. This makes the exploration of the sequence pattern space much more efficient.

Candidate Generation

According to Apriori property, candidate patterns of length ≥ 2 can be obtained by concatenating shorter frequent patterns. For each ordered pair (p_1, p_2) where p_1 and p_2 are two length-l frequent patterns, we create a length-$(l+1)$ candidate pattern if (1) p_1 excluding the first timestamp, matches exactly with p_2 excluding the last timestamp; and (2) the first timestamp in p_1 is earlier than the last timestamp in p_2. A candidate length-$(l+1)$ pattern is generated by concatenating p_1 with the last element of p_2.

3 Community Trend Outlier Detection

In this section, we will discuss how to exploit set P of frequent soft patterns obtained after pattern extraction (Sec. 2) to assign an outlier score to each sequence in the dataset. When capturing evolutionary trends, length-1 patterns are not meaningful, as they are defined on single snapshots. So, we remove them from set P. Although the input sequences are all of length T, each pattern could be of any length $\leq T$ and could be gapped. While a sequence represents a point distribution at each timestamp, a pattern represents a cluster centroid for each timestamp. Each cluster is associated with a support, and there might be other statistics to describe the cluster, such as maximum distance of any point from the centroid and average distance of all points within cluster from the centroid. Intuitively, patterns consisting of compact clusters (clusters with low average distances) with high support are the most important for outlier detection.

Outlier Detection Problem

Input: Set P of frequent soft patterns with support $\geq$ *min_sup*.

Output: Set of *CTOutlier* objects.

3.1 Pattern Configurations and Best Matching Pattern

A non-outlier object may follow only one frequent pattern while deviating from all other patterns. Hence, it is incorrect to compute outlier score for an object by adding up its outlierness with respect to each pattern, weighted by the pattern support. Also, it is not meaningful to compute outlier score just based on the best matching pattern. The reason is that often times, even outlier sequences will follow some length-2 pattern; but such a short pattern does not cover the entire length of the sequence. Therefore, we propose to analyze the outlierness of a sequence with respect to its different projections by dividing the pattern space into different configurations.

Configuration: A *configuration* c is simply a set of timestamps with size ≥ 2. Let P_c denote the set of patterns corresponding to the configuration c.

Finding Best Matching Pattern

A normal sequence generally follows a particular trend (frequent pattern) within every configuration. A sequence may have a very low match with most patterns but if it matches completely with even one frequent pattern, intuitively it is not an outlier. (Here we do not consider the situation of group outliers, where all sequences following a very different pattern could be called outlier.) Hence, we aim at finding the pattern which matches the sequence the most for that configuration. Note that patterns corresponding to big loose clusters match a large number of sequences, and thus we should somehow penalize such patterns over those containing compact clusters.

Based on the principles discussed above, we design the following matching rules. Let a pattern p be divided into two parts ϕ_{po} and θ_{po}. ϕ_{po} (θ_{po}) consists of the set of timestamps where sequence for object o and pattern p match (do not match) each other. E.g., considering pattern $p =$ ▶ and sequence $o =$ ★, $\theta_{po} = \{1\}$ and $\phi_{po} = \{2, 3\}$.

Match Score: We define the match score between an object o and a pattern p as follows.

$$match(p,o) = \sum_{t \in \phi_{po}} \frac{sup(P_{t_p}) \times sup(P_{t_p}, S_{t_o})}{avgDist(P_{t_p})} \tag{3}$$

where $avgDist(P_{t_p})$ is the average distance between the objects within the cluster and the cluster centroid P_{t_p}, and $sup(P_{t_p}, S_{t_o}) = 1 - \frac{Dist(P_{t_p}, S_{t_o})}{maxDist(P_{t_p})}$. This definition is reasonable because the score is higher if (1) the object and the pattern match for more timestamps; (2) pattern has higher support; (3) pattern contains compact clusters; and (4) object lies closer to the cluster centroid across various timestamps.

Best Matching Pattern: The best matching pattern bmp_{co} is the pattern $p \in P_c$ with the maximum match score $match(p,o)$. In the toy example, the best matching pattern for sequence ★ with respect to configuration $\{1,2,3\}$ is ▶.

3.2 Outlier Score Definition

Given a sequence, we first find the best matching pattern for every configuration and then define the outlier score as the sum of the scores of the sequence with respect to each configuration. The outlier score of object o is thus expressed as:

$$outlierScore(o) = \sum_{c=1}^{|C|} outlierScore(c,o) = \sum_{c=1}^{|C|} outlierScore(bmp_{co}, o) \tag{4}$$

where bmp_{co} is the best matching pattern for configuration c and object o, and C is the set of all configurations.

Let $\tilde{p}$ denote the best matching pattern bmp_{co} in short. Then we can express the mismatch between $\tilde{p}$ and o by $\sum_{t \in \theta_{\tilde{p}o}} sup(P_{t_{\tilde{p}}}) \times \frac{Dist(P_{t_{\tilde{p}}}, S_{t_o})}{maxDist(P_{t_{\tilde{p}}})}$. Thus, mismatch between the pattern and the soft sequence for o is simply the timestamp-wise mismatch weighted by the support of pattern at that timestamp. Finally, the importance of the pattern is captured by multiplying this mismatch score by the overall support of the pattern. As can be seen, $outlierScore(\tilde{p}, o)$ as expressed in Eq. 5 takes into account the support of the pattern, number of mismatching timestamps, and the degree of mismatch.

$$outlierScore(bmp_{co}, o) = outlierScore(\tilde{p}, o) = sup(\tilde{p}) \times \sum_{t \in \theta_{\tilde{p}o}} sup(P_{t_{\tilde{p}}}) \times \frac{Dist(P_{t_{\tilde{p}}}, S_{t_o})}{maxDist(P_{t_{\tilde{p}}})} \tag{5}$$

Time Complexity Analysis

Finding best matching patterns for all sequences takes $O(N|P|TK)$ time where $|P|$ is the number of patterns. Number of configurations $|C| = 2^T - T - 1$. So, outlier score computation using the best matching patterns takes $O(N(2^T - T - 1)KT)$ time where K is the maximum number of clusters at any timestamp. Thus, our outlier detection method is $O(NTK(2^T - T - 1 + |P|))$ in time complexity, i.e., linear in the number of objects. Generally, for real datasets, T is not very large and so the complexity is acceptable. For larger T, one may use sampling from the set of all possible configurations, rather than using all configurations. Our results (Sec. 4) show that considering only length-2 ungapped configurations can also provide reasonable accuracy.

Note that we did not include the pattern generation time in the time complexity analysis. This is because it is difficult to analyze time complexity of algorithms using Apriori pruning. However it has been shown earlier that Apriori techniques are quite efficient for pattern generation [5].

3.3 Summing Up: *CTOutlier* Detection Algorithm (CTODA)

The proposed *CTOutlier* Detection Algorithm (Algo. 1) can be summarized as follows. Given a dataset with N soft sequences each defined over T timestamps, soft patterns are first discovered from Steps 1 to 12 and then outlier scores are computed using Steps 13 to 20.

Algorithm 1. *CTOutlier* Detection Algorithm (CTODA)

Input: (1) Soft sequences for N objects and T timestamps (represented using matrix S). (2) Minimum Support: min_sup.
Output: Outlier Score for each object.

```
 1: Set of frequent patterns $P \leftarrow \phi$                                  ▷ Pattern Extraction
 2: Let $L_l$ be the set of length-l frequent patterns. $\{L_l\}_{l=1}^{T} \leftarrow \phi$.
 3: Let $C_l$ be the set of length-l candidate patterns. $\{C_l\}_{l=1}^{T} \leftarrow \phi$.
 4: for each timestamp $t$ do
 5:     $C_1 \leftarrow$ Cluster $S_t$ (i.e., part of $S$ for timestamp $t$).
 6:     $L_1 \leftarrow L_1 \cup \{f | f \in C_1$ and $sup(f) \geq min\_sup\}$
 7: end for
 8: for l=2 to T do
 9:     $C_l \leftarrow getCandidates(L_{l-1})$.
10:     $L_l \leftarrow \{f | f \in C_l$ and $sup(f) \geq min\_sup\}$.
11:     $P \leftarrow P \cup L_l$.
12: end for
13: $C \leftarrow$ Set of configurations for $T$ timestamps.                     ▷ Outlier Detection
14: for each object $o$ do
15:     for each configuration $c \in C$ do
16:         Compute the best matching pattern $\tilde{p} \in P$ for object $o$ and configuration $c$ using Eq. 3.
17:         Compute $outlierScore(\tilde{p}, o)$ using Eq. 5.
18:     end for
19:     Compute $outlierScore(o)$ using Eq. 4.
20: end for
```

Next, we discuss two important practical issues in implementing the proposed community trend outlier detection algorithm.

Effect of Varying *min_sup*

min_sup decides the number of patterns discovered, given a temporal dataset. Higher *min_sup* implies that some patterns may not get detected and hence even non-outlier sequences may get marked as outliers. However, their outlier scores will still be lower than the scores of extreme outliers because they are closer to the cluster centroids for each individual timestamp. Also, the number of configurations for which normal sequences deviate from patterns, will be smaller than the number of configurations for outlier sequences. However, a very high *min_sup* might mean that no patterns get discovered for a lot of configurations.

In that case, many sequences might be assigned same high outlier scores, which is undesirable.

If *min_sup* is too low, then the discovered patterns may represent an overfitted model of the data. Thus, even outliers may get modeled as normal patterns, and then we may not be able to discover many outliers. Also a lower value of *min_sup* will generate a bigger set of patterns so that the overall pattern generation may become very inefficient with respect to time and memory.

Therefore, there is a tradeoff between lower and higher *min_sup*. The best way to select *min_sup* is to determine what percentage of sequences can be considered to represent a significant pattern. This could be quite domain dependent. In some domains, it might be completely fine even if a very few objects demonstrate a pattern while in other domains, one might need to use a larger *min_sup* value.

Hierarchical Clustering

In this paper, we performed single-level clustering of the distributions corresponding to the community detection results. However, one can also perform hierarchical clustering. Multi-level soft patterns discovered using such a hierarchical clustering per snapshot, could be more expressive. Using DBLP example again, one may be able to express that a sub-area in timestamp 1 (lower level cluster) evolved into a mature research area in timestamp 2 (higher level cluster). We plan to explore the benefits of using such hierarchical clustering methods as part of future work.

4 Experiments

Evaluation of outlier detection algorithms is quite difficult due to lack of ground truth. We generate multiple synthetic datasets by injecting outliers into normal datasets, and evaluate outlier detection accuracy of the proposed algorithms on the generated data. We also conduct case studies by applying the method to real data sets. We perform comprehensive analysis to justify that the top few outliers returned by the proposed algorithm are meaningful. The code and the data sets are available at: `http://blitzprecision.cs.uiuc.edu/CTOutlier`

4.1 Synthetic Datasets

Dataset Generation

We generate a large number of synthetic datasets to simulate real evolution scenarios, each of which consists of 6 timestamps. We first create a dataset with normal points and then inject outliers. The accuracy of the algorithms is then measured in terms of their effectiveness in discovering these outliers. For each dataset, we first select the number of objects (N), the number of full-length (i.e., length=6) patterns ($|F|$), the percentage of outliers (Ψ) and the outlier degree (γ). Next, we randomly select the number of clusters per timestamp. Each cluster is represented by a Gaussian distribution with a fixed mean and variance. Figure 2 shows the clusters with their 2σ boundaries. For each full pattern, we first choose a cluster at each timestamp and then select a Gaussian distribution with small variance within the cluster. Once we have fixed the Gaussian distribution to be used, we generate $N/|F|$ points per timestamp for the pattern. Each full-length pattern results into $2^6 - 6 - 1 = 57$ patterns. We ensure that each cluster is part of at least one pattern. Once patterns are generated, we generate outliers as follows. For every outlier, first we select the base pattern for the outlier. An outlier follows the base pattern for $\lceil T \times \gamma \rceil$ timestamps and deviates from the pattern for the remaining timestamps. We fix a set of $\lceil T \times \gamma \rceil$ timestamps and randomly select a cluster different from the one in the pattern for the remaining timestamps. Figure 2 shows the first 4 timestamps (out of 6 – for lack of space) of a dataset created with N=1000, $|P|$=570 ($|F|$=10), Ψ=0.5 and γ=0.6. Colored points are normal points following patterns while larger black shapes (■, ▼, ◀, ▲, ⧫) are the injected outliers. For example, the outlier (▼) usually belongs to the ★ pattern, except for the third timestamp where it switches to the yellow ▶ pattern.

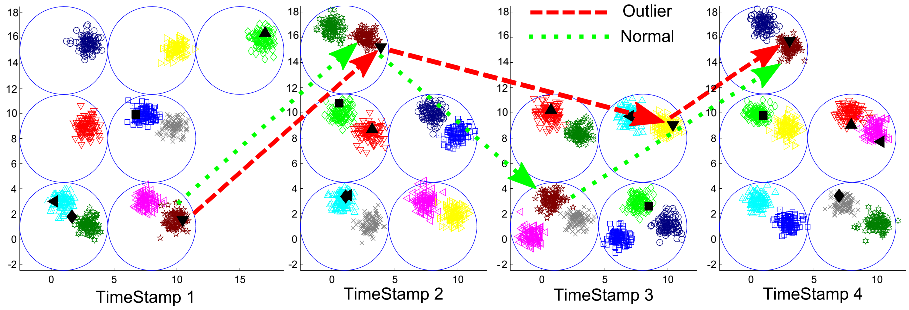


Fig. 2. First Four Snapshots of our Synthetic Dataset

Table 1. Synthetic Dataset Results (*CTO*=The Proposed Algorithm *CTODA*, *BL1*=Consecutive Baseline, *BL2*=No-gaps Baseline) for Outlier Degree=0.5 and 0.8, i.e., Outliers follow Base Pattern for 3/6 and 5/6 Timestamps respectively.

N	Ψ	$\lvert\gamma\rvert$ = 0.5									$\lvert\gamma\rvert$ = 0.8								
		$\lvert F\rvert$ = 5			$\lvert F\rvert$ = 10			$\lvert F\rvert$ = 15			$\lvert F\rvert$ = 5			$\lvert F\rvert$ = 10			$\lvert F\rvert$ = 15		
	(%)	CTO	BL1	BL2	CTO	BL1	BL2	CTO	BL1	BL2	CTO	BL1	BL2	CTO	BL1	BL2	CTO	BL1	BL2
1000	1	**95.0**	92	89	**92.5**	86	90	**93.5**	83	92	**95.5**	85.5	92	83	76.5	**84.0**	**92.0**	77	86.0
	2	**98.0**	94.2	95.5	**94.0**	88.2	92	**95.5**	87.2	93.2	**98.2**	94.5	96.5	**91.2**	86.5	90	**95.5**	76	94.0
	5	**99.5**	96.8	97.4	**96.5**	95.3	96.2	**97.9**	93.1	97.1	**99.0**	95.7	97.3	**96.3**	91	95.9	**97.4**	79.3	96.7
5000	1	**97.0**	90.6	91.5	**91.9**	86	89.4	**91.8**	84.3	89.9	**95.8**	83.5	89.8	**84.4**	76.6	84.4	**88.4**	73.1	86.1
	2	**97.2**	92	92.8	**94.0**	91.2	93	**96.4**	89	94	**97.9**	89.6	94	**89.4**	85.6	88.4	**95.4**	79.8	93.1
	5	**99.4**	96.9	97.3	**96.3**	94.7	96.3	**97.6**	91	96.3	**98.8**	95.4	97.6	**95.0**	90.5	94.7	**97.7**	79.7	96.9
10000	1	**97.4**	90	90.4	**90.8**	85.4	88.1	**92.8**	84.5	88.2	**95.6**	84.2	89.5	81.8	76.4	**82.8**	**91.8**	76.5	87.6
	2	**98.2**	91.6	92.6	**93.2**	90.5	92.7	**95.0**	89.3	92.4	**98.0**	91.1	95	89.9	86.9	**90.7**	**95.8**	80.6	93.3
	5	**99.0**	96.8	97.1	**96.2**	94.4	96.2	**97.9**	89.6	96.8	**99.1**	95.8	98	**95.3**	90.1	95.3	**97.3**	76.4	96.6

Results on Synthetic Datasets

We experiment using a variety of settings. We fix the minimum support to $(100/|F| - 2)\%$. For each setting, we perform 20 experiments and report the mean values. We compare with two baselines: *Consecutive (BL1)* and *No-gaps (BL2)*. Often times evolution is studied by considering pairs of consecutive snapshots of a dataset, and then integrating the results across all pairs. One can use such a method to compute outliers across each pair of consecutive snapshots and then finally combine results to get an outlier score across the entire time duration. We capture this in our *BL1* baseline. Note that we consider only those configurations which are of length-2 and which contain consecutive timestamps in this baseline. Thus, *BL1* will mark an object as outlier if its evolution across consecutive snapshots is much different from observed length-2 patterns (with no gaps). For the *BL2* baseline, we consider all configurations corresponding to patterns of arbitrary length without any gaps. Note that this baseline simulates the trajectory outlier detection scenario with respect to our framework. Recall that our method is named as *CTODA*.

We change the number of objects from 1000 to 5000 to 10000. We vary the percentage of outliers injected into the dataset as 1%, 2% and 5%. The outlier degree is varied as 0.5, 0.6 and 0.8 (i.e., 3, 4 and 5 timestamps). Finally, we also use different number of full-length patterns ($|F| = 5, 10, 15$), i.e., $|P| = 285, 570, 855$, to generate different datasets. Table 1 shows the results in terms of precision when the number of retrieved outliers equals to the actual number of injected outliers for γ=0.5 and 0.8. Average standard deviations are 3.11%, 4.85% and 3.39% for *CTODA* and the two baselines respectively. Results with γ=0.6 are also similar; we do not show them here for lack of space. On an average *CTODA* is 7.4% and 2.3% better than the two baselines respectively.

The reasons why the proposed *CTODA* method is superior are as follows. Consider a soft sequence $\langle S_{1_o}, S_{2_o}, S_{3_o}\rangle$. Both the soft patterns $\langle S_{1_o}, S_{2_o}\rangle$ and $\langle S_{2_o}, S_{3_o}\rangle$ might be frequent while $\langle S_{1_o}, S_{2_o}, S_{3_o}\rangle$ might not be frequent. This can happen if the sequences which have the pattern $\langle S_{1_o}, S_{2_o}\rangle$ and the sequences with the pattern $\langle S_{2_o}, S_{3_o}\rangle$ are disjoint. This case clearly shows why our method which computes support for patterns of arbitrary lengths is better than baseline *BL1* which considers only patterns of length two with consecutive timestamps. Now let us show that gapped patterns can be beneficial even when we consider contiguous patterns of arbitrary lengths. Consider two soft gapped patterns $p = \left\langle P_{1_p}, P_{2_p}, P_{4_p}\right\rangle$ and $q = \left\langle P_{1_q}, P_{3_q}, P_{4_q}\right\rangle$ such that $P_{1_p} = P_{1_q}$ and $P_{4_p} = P_{4_q}$. Now p might be frequent while q is not. However, this effect cannot be captured if we consider only contiguous patterns. This case thus shows why our approach is better than *BL2*.

We ran our experiments using a 2.33 GHz Quad-Core Intel Xeon processor. On an average, the proposed algorithm takes 83, 116 and 184 seconds for N=1000, 5000 and 10000 respectively. Of this 74, 99 and 154 seconds are spent in pattern generation while the remaining time is spent in computing outliers given these patterns.

4.2 Real Datasets

Dataset Generation

We perform experiments using three real datasets: *GDP*, *Budget* and *Four Area* (subset of *DBLP*).

***GDP*:** The *GDP* dataset consists of (Consumption, Investment, Public Expenditure and Net Exports) components for 89 countries for 1982-91 (10 snapshots)[1].

***Budget*:** The *Budget* dataset consists of (Pensions, Health Care, Education, Defense, Welfare, Protection, Transportation, General Government, Other Spending) components for 50 US states for 2001-10 (10 snapshots)[2].

***Four Area*:** This is a subset of DBLP for the 4 areas of data mining (DM), databases (DB), information retrieval (IR) and machine learning (ML) and consists of papers from 20 conferences (5 per area). For details, read [12]. We obtain 5 co-authorship snapshots corresponding to the years 2000-01 to 2008-09 for 643 authors.

Results on Real Datasets

CTOutliers are objects that break many temporal community patterns. We will provide a few interesting case studies for each dataset and explain the intuitions behind the identified outliers on how they deviate from the best matching patterns.

***GDP*:** We find 3682 patterns when minimum support is set to 20% for the 89 countries. The top five outliers discovered are Uganda, Congo, Guinea, Bulgaria and Chad, and we provide reasonings about Uganda and Congo as examples to support our claims as follows. National Resistance Army (NRA) operating under the leadership of the current president, Yoweri Museveni came to power in 1985-86 in **Uganda** and brought reforms to the economic policies. Uganda showed a sudden change of (45% consumption and 45% net exports) to (80% consumption and 1-2% net exports) in 1985. Unlike Uganda, other countries like Iceland, Canada, France with such ratios of consumption and net export maintained to do so. Like many other countries, **Congo** had 45-48% of its GDP allocated to consumption and 36-42% of GDP for government expenditure. But unlike other countries with similar pattern, in 1991, consumption decreased (to 29% of GDP) but government expenditure increased (56% of GDP) for Congo. This drastic change happened probably because opponents of the then President of Congo (Mobutu Sese Seko) had stepped up demands for democratic reforms.

***Budget*:** We find 41545 patterns when minimum support is set to 20% for the 50 states. The top five outliers discovered are AK, DC, NE, TN and FL, and the case on AK is elaborated as follows. For states with 6% pension, 16% healthcare, 32% education, 16% other spending in 2006, it has been observed that healthcare increased by 4-5% in 2009-10 while other spending decreased by 4%. However,

[1] `http://www.economicswebinstitute.org/ecdata.htm`

[2] `http://www.usgovernmentspending.com/`

in the case of **Arkansas (AK)** which followed a similar distribution in 2006, healthcare decreased by 3% and other spending increased by 5% in 2009-10. More details can be found in Fig. 3. The right figure shows the distribution of expenses for AK for the 10 years, while the left part shows similar distribution averaged over 5 states which have a distribution very similar to AK for the years 2004-09. One can see that Arkansas follows quite a different distribution compared to the five states for other years especially for 2002.

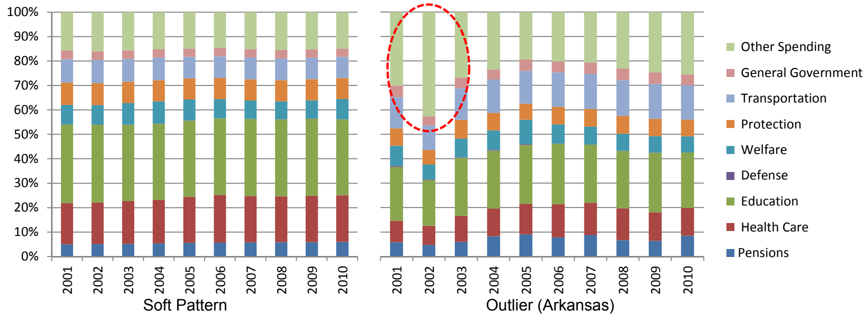


Fig. 3. Average Trend of 5 States with Distributions close to that of AK for 2004-09 (Left – Pattern), Distributions of Budget Spending for AK (Right – Outlier)

***Four Area*:** We perform community detection on each snapshot of the original co-authorship dataset using *EvoNetClus* [23] to obtain soft sequences. This leads to 8, 7, 9, 7, 7 clusters for the five snapshots respectively. We find 1008 patterns when minimum support is set to 10%. The top few outliers discovered are Hakan Ferhatosmanoglu, Mehul A. Shah, and Arbee L. P. Chen. The general trends observed are authors switching between DM and ML areas, or switching between IR and DB. However, research areas associated with **Hakan Ferhatosmanoglu** demonstrate a special combination of different research areas. The soft sequence associated with him looks as follows. $\langle 2000-01 : (IR : 0.75, DB : 0.25), 2002-03 : (IR : 1), 2004-05 : (DB : 1), 2006-07 : (DB : 0.67, DM : 0.33), 2008-09 : (DB : 0.5, ML : 0.5)\rangle$. He started out as a researcher in IR and has changed focus to DM and then ML. So, clearly he does not fit into any of the trends and hence is an interesting outlier. Similar observations can be found for the other detected outliers.

In summary, the proposed algorithm is highly accurate in identifying injected outliers in synthetic datasets and it is able to detect some interesting outliers from each of the real datasets.

5 Related Work

Outlier (or anomaly) detection [10,18] is a very broad field and has been studied in the context of a large number of application domains. Outliers have been discovered in high-dimensional data [1], uncertain data [3], stream data [4], network data [13] and time series data [11].

Temporal Outlier Detection

Recently, there has been significant interest in detecting outliers from evolving datasets [4,14,15], but none of them explores the outliers with respect to communities in a general evolving dataset. Outlier detection methods for data streams [2,7] have no notion of communities. *CTOutlier* detection could be considered as finding outlier trajectories across time, given the soft sequence data. However as discussed in Sec. 1, there are major differences, making trajectory outlier detection techniques unsuitable for the task.

Community Outlier Detection

Community outlier detection has been studied for a static network setting [13] or for a setting of two network snapshots [17], but we develop an algorithm for a general evolving dataset with multiple snapshots. Group (community) identification in evolving scenarios has been studied traditionally in the context of hard patterns [16,20], while we discover soft patterns capturing *subtle community* evolution trends.

Thus, though significant literature exists both for temporal as well as community outlier detection, we discover novel outliers by considering both the temporal and the community dimensions together.

6 Conclusions

In datasets with continuously evolving values, it is very important to detect objects that do not follow normal community evolutionary trend. In this paper, we propose a novel concept of an outlier, denoting objects that deviate from temporal community norm. Such objects are referred to as Community Trend Outliers (*CTOutliers*), and are of great practical importance to numerous applications. To identify such outliers, we proposed an effective two-step outlier detection algorithm *CTODA*. The proposed method first conducts soft pattern mining efficiently and then detects outliers by measuring the objects' deviations from the normal patterns. Extensive experiments on multiple synthetic and real datasets show that the proposed method is highly effective and efficient in detecting meaningful community trend outliers. In the future, we plan to further our studies on evolutionary outlier detection by considering various evolution styles in different domains.

Acknowledgements. Research was sponsored in part by the Cyber Security project (W911NF-11-2-0086) and NSCTA project (W911NF-09-2-0053) of U.S. Army Research Laboratory, NSF IIS-0905215, MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC and U.S. Air Force Office of Scientific Research MURI award FA9550-08-1-0265. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. Aggarwal, C.C., Yu, P.S.: Outlier Detection for High Dimensional Data. SIGMOD Records 30, 37–46 (2001)
2. Aggarwal, C.C.: On Abnormality Detection in Spuriously Populated Data Streams. In: SDM, pp. 80–91 (2005)
3. Aggarwal, C.C., Yu, P.S.: Outlier Detection with Uncertain Data. In: SDM, pp. 483–493 (2008)
4. Aggarwal, C.C., Zhao, Y., Yu, P.S.: Outlier Detection in Graph Streams. In: ICDE, pp. 399–409 (2011)
5. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: VLDB, pp. 487–499 (1994)
6. Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V.: Discovering Clusters in Motion Time-Series Data. In: CVPR, pp. 375–381 (2003)
7. Angiulli, F., Fassetti, F.: Detecting Distance-based Outliers in Streams of Data. In: CIKM, pp. 811–820 (2007)
8. Basharat, A., Gritai, A., Shah, M.: Learning Object Motion Patterns for Anomaly Detection and Improved Object Detection. In: CVPR, pp. 1–8 (2008)
9. Bernecker, T., Kriegel, H.P., Renz, M., Verhein, F., Zuefle, A.: Probabilistic Frequent Itemset Mining in Uncertain Databases. In: KDD, pp. 119–128 (2009)
10. Chandola, V., Banerjee, A., Kumar, V.: Anomaly Detection: A Survey. ACM Computing Surveys 41(3) (2009)
11. Fox, A.J.: Outliers in Time Series. Journal of the Royal Statistical Society. Series B (Methodological) 34(3), 350–363 (1972)
12. Gao, J., Liang, F., Fan, W., Sun, Y., Han, J.: Graph-based Consensus Maximization among Multiple Supervised and Unsupervised Models. In: NIPS, pp. 585–593 (2009)
13. Gao, J., Liang, F., Fan, W., Wang, C., Sun, Y., Han, J.: On Community Outliers and their Efficient Detection in Information Networks. In: KDD, pp. 813–822 (2010)
14. Ge, Y., Xiong, H., Hua Zhou, Z., Ozdemir, H., Yu, J., Lee, K.C.: Top-Eye: Top-K Evolving Trajectory Outlier Detection. In: CIKM, pp. 1733–1736 (2010)
15. Ghoting, A., Otey, M.E., Parthasarathy, S.: LOADED: Link-Based Outlier and Anomaly Detection in Evolving Data Sets. In: ICDM, pp. 387–390 (2004)
16. Gudmundsson, J., Kreveld, M., Speckmann, B.: Efficient Detection of Motion Patterns in Spatio-temporal Data Sets. In: GIS, pp. 250–257 (2004)
17. Gupta, M., Gao, J., Sun, Y., Han, J.: Integrating Community Matching and Outlier Detection for Mining Evolutionary Community Outliers. In: KDD (to appear, 2012)
18. Hodge, V.J., Austin, J.: A Survey of Outlier Detection Methodologies. AI Review 22(2), 85–126 (2004)
19. Lee, J.-G., Han, J., Li, X.: Trajectory Outlier Detection: A Partition-and-Detect Framework. In: ICDE, pp. 140–149 (2008)
20. Li, Z., Ding, B., Han, J., Kays, R.: Swarm: Mining Relaxed Temporal Moving Object Clusters. In: VLDB, pp. 723–734 (2010)
21. Muzammal, M., Raman, R.: Mining Sequential Patterns from Probabilistic Databases. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS, vol. 6635, pp. 210–221. Springer, Heidelberg (2011)
22. Pelleg, D., Moore, A.W.: X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In: ICML, pp. 727–734 (2000)
23. Sun, Y., Tang, J., Han, J., Gupta, M., Zhao, B.: Community Evolution Detection in Dynamic Heterogeneous Information Networks. In: KDD-MLG, pp. 137–146 (2010)

Data Structures for Detecting Rare Variations in Time Series

Caio Valentim, Eduardo S. Laber, and David Sotelo

Departamento de Informática, PUC-Rio, Brazil

Abstract. In this paper we study, from both a theoretical and an experimental perspective, algorithms and data structures to process queries that help in the detection of rare variations over time intervals that occur in time series. Our research is strongly motivated by applications in financial domain.

1 Introduction

The study of time series is motivated by applications that arise in different fields of human knowledge such as medicine, physics, meteorology and finance, just to cite a few. For some applications, involving time series, general purpose spreadsheets and databases provide a good enough solution to analyse them.

However, there are applications in which the time series are massive as in the analysis of data captured by a sensor in a milliseconds basis or in the analysis of a series of quotes and trades of stocks in an electronic financial market. For these series, traditional techniques for storing data may be inadequate due to its time/space consumption. This scenario motivates the research on data structures and algorithms to handle massive time series [13].

The focus of our research is to develop data structures that allow the identification of variations on time series that are rare, i.e., that occur just a few times over a long time period.

Let $A = (a_1, ..., a_n)$ be a time series with n values and let a time index be an integer belonging to the set $\{1, \ldots, n\}$. We use two positive numbers, t and d, to capture variations in the time series A during a time interval. More precisely, we say that a pair of time indexes (i, j) of A is a (t, d)-event if $0 < j - i \leq t$ and $a_j - a_i \geq d$.

We want to design efficient algorithms/data structures to handle the following queries.

- $AllPairs(t, d)$. This query returns all (t, d)-events in time series A.
- $Beginning(t, d)$. This query returns all time indexes i such that there is a time index j for which (i, j) is a (t, d)-event in time series A.

The reason why we focus on the above queries is because we believe that they are basic queries for the study of rare variations in time series so that the techniques developed to address them could be extended to many other queries with the same flavor.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 709–724, 2012.

To be more concrete, we describe an application where an efficient processing of our queries turns out to be useful. The prices of two stocks with similar characteristics (e.g. belonging to the same segment of the industry) usually have a strong correlation. A very popular trading strategy in financial stock markets is to estimate an expected ratio between the prices of two similar stocks and then bet that the observed ratio will return to the estimated one when it significantly deviates from it [8].

A strategy designer may be interested to study if rare variations between the ratio of two stocks tend to revert to its mean (normal value). For that he(she) may specify a possibly long list of pairs (t, d) and then study the behavior of the time series right after the occurrence of a (t, d)-event, for each pair (t, d) in the list. A naive scan over a time series with the aim of looking for (t, d)-events, for a fixed pair (t, d), requires $O(nt)$ time. In some stock markets (e.g. NYSE), where over 3000 stocks are daily negotiated, and a much larger number of pairs of stocks can be formed, a naive approach turns out to be very expensive. Thus, cheaper alternatives to detect (t, d)-events may be desirable.

Still in this context, given a the length t of a time interval and a threshold p, one may ask for the minimum d for which the number of (t, d)-events is at most p, that is, (t, d)-events are rare. If we want to process this kind of query for many time intervals t and many thresholds p, an efficient processing of our primitive queries is also required.

To make the presentation easier we focus on positive variations ($d > 0$) in our queries. However, all the techniques developed here also apply for processing negative variations. The following notation will be useful for our discussion. The Δ-value of a pair (i, j) is given by $j - i$ and its deviation is given by $a_j - a_i$. We say that i is the startpoint of a pair (i, j) while j is its endpoint.

Our Contributions. We develop and analyze algorithms and data structures to process queries AllPairs(t, d) and Beginning(t, d). The relevant parameters for our analysis are the time required to preprocess the data structure, its space consumption and the query elapsed time.

To appreciate our contributions it is useful to discuss some naive approaches to process our queries.

A naive algorithm to process $AllPairs(t, d)$ scans the list A and for each startpoint i, it reports the time indexes j in the interval $[i + 1, i + t]$ for which $a_j - a_i \geq d$. This procedure handles query $AllPairs(t, d)$ in $O(nt)$ time. A similar procedure also process $Beginning(t, d)$ in $O(nt)$ time.

Another simple approach to process $Beginning(t, d)$ is to use a table T, indexed from 1 to $n - 1$; the entry $T[t]$, for each t, has a pointer to a list containing the pairs in the set $\{(1, m_1), (2, m_2), \ldots, (n - 1, m_{n-1})\}$, where m_i is the time index of the largest value among $a_{i+1}, a_{i+2}, \ldots, a_{i+t}$. Each of these $n - 1$ lists is sorted by decreasing order of the deviations of their pairs. To process Beginning(t, d), we scan the list associated with $T[t]$ and report the startpoint of every pair that is a (t, d)-event. The scan is aborted when a pair that has deviation smaller than d is found. The procedure process $Beginning(t, d)$ in $O(k + 1)$ time, where k is the size of the solution set. Along these lines, it is not difficult to design a data structure that also handles query $\mathit{AllPairs}(t, d)$ in $O(1+k)$ time. Though this kind of approach achieves optimal querying time, its space consumption could be prohibitive for most of practical applications since we have to store $\Theta(n^2)$ pairs.

The solutions we propose/analyze in this paper make an intensive use of data structures that support range minimum(maximum) queries(RMQ's). Let V be a vector of real entries, indexed from 1 to n. A range minimum(maximum) query (RMQ) over V, with input range $[i, j]$, returns the index of the entry of V with minimum(maximum) value, among those with indexes in the range $[i, j]$. This type of query is very well understood by the computational geometry community [1,6]. In fact, there are known data structures of size $O(n)$ that support RMQs in constant time.

A simple but crucial observation is that the queries $\mathit{AllPairs}(t, d)$ and $\mathit{Beginning}(t, d)$ can be translated into multiple RMQ's (details are given in the next section). Thus, by using data structures that support RMQ's, we can handle both queries in $O(n + k)$ time, where k is the size of the solution set. In fact, $\mathit{Beginning}(t, d)$ can be handled with the same time complexity without any preprocessing as we explain in Section 4.1. These solutions have the advantage of being very space efficient, they require $O(n)$ space. Their main drawback, however, is that they are not output sensitive in the sense that they require $\Omega(n)$ time regardless to the size of the solution set.

When the list of (t, d) pairs for which we are interested to find/study the (t, d)-events is long, it may be cheaper, from a computational point of view, to preprocess the time series into a data structure and then handling the queries. However, the design of a compact data structure that allows efficient query processing seems to be challenging task. The key idea of our approach is to index a set of specials pairs of time indexes rather than all possible pairs. We say that a pair of time indexes (i, j) is <u>special</u> with respect to time series A if the following conditions hold: $i < j$, $a_i < \min\{a_{i+1}, \ldots, a_j\}$ and $a_j > \max\{a_i, \ldots, a_{j-1}\}$. If a (t, d)-event is also a special pair we say that it is a special (t, d)-event.

Let S be the set of special pairs of the time series A. We propose a data structure that requires $O(|S| + n)$ space/preprocessing time and handles query $\mathit{AllPairs}(t, d)$ in $O(k+1)$ time, where k is the size of the solution set. In addition, we propose a structure that requires $O(|S| \log |S|)$ space/preprocessing time and process $\mathit{Beginning}(t, d)$ in $O(\log n + f(\log f + \log t) + k)$ time, where f is the number of distinct time indexes that are endpoints of special (t, d)-events and k is the number of startpoints to be reported. Because of the symmetry between startpoints and endpoints and the fact that f is the number of endpoints for a restricted class of (t, d)-events(the special ones), we expect f to be smaller than k, that is, we expect to pay logarithmic time per solution reported. In fact, our experimental results are in accordance with this hypothesis.

The preprocessing time/space consumption of some of our solutions heavily depend on the number of special pairs of the time series under consideration. Thus, we investigate the expected value of this quantity. First, we prove that the expected number of special pairs of a random permutation of $\{1, \ldots, n\}$, is $n - \ln n$ with high probability. This result is interesting in the sense that the number of special pairs of a time series with distinct values is equal to the number of special pairs of the permutation of $\{1, \ldots, n\}$ in which the time series can be naturally mapped on. In addition, we evaluated the number of special pairs for 96 time series consisting of stock prices, sampled in a minute basis, over a period of three years, from the Brazilian stock market. We observed that the number of special pairs is, in average, 2.5 times larger than the size of the corresponding time series.

Finally, we performed a set of experiments to evaluate the performance of our algorithms. These experiments confirm our theoretical results and also reveal some interesting aspects of our solutions that are not explicited by the underlying theory.

Related Work. In the last decade, a considerable amount of research has been carried on to develop data mining techniques for time series as thoroughly discussed in a recent survey by Fu [7]. An important research sub-field pointed out by this survey asks for how to identify a given pattern (subsequence matching) in the time series [5,12,14,10]. Though the problem studied here is related with this line of research, our queries do not fit in the framework proposed in [5] so that the available techniques do not seem to be adequate to handle them. In fact, one single query (e.g. $Beginning(t, d)$) that we deal with could be translated into a family of queries involving subsequences.

Our work is also closely related to the line of research that focus on identifying technical patterns in financial time series [11,14]. Technical patterns as head &shoulders, triangles, flags, among others, are shapes that are supposed to help in forecasting the behaviour of stocks prices. Here we focus on a specific type of pattern that is useful for the study of mean reversion processes that occur in financial markets [3].

The problem of developing data structures to support Range Minimum (Maximum) Queries (RMQ problem), as previously explained, is closely related to our problems. The first data structure of linear size that can be constructed in linear time and answer RMQ queries in constant time was discovered by Harel and Tarjan [9]. Their structure, while a huge improvement from the theoretical side, was difficult to implement. After this paper, some others were published proposing some simpler, though still complex, data structures. A work from Bender et al.[1] proposed the first uncomplicated solution for the RMQ problem. The work by Fischer and Heun[6] improves the solution by Bender et. al. and also present an experimental study that compares some available strategies.

Finally, our problem has some similarities to a classical computational geometry problem, named the fixed-radius near neighbors problem. Given a set of points n in a m-dimensional Euclidean space and a distance d, the problem consists of reporting all pairs of points within distance not greater than d. Bentley et al [2] introduced an algorithm that reports all k pairs that honor this property in $O(3^m n + k)$. The same approach can be applied if a set $\{d_1, d_2, \ldots, d_m\}$ of maximum distances is given, one for each dimension. However, it is not clear for us if similar ideas can be applied to generate all pairs within distance *not* *less* than d or, as in the case of our problem, not greater than d_1 in the first dimension and not less than d_2 in the second.

Paper Organization. Section 2 introduces some basic concepts/notations that will be used to explain our approach. In Section 3 and 4, we present, respectively, our data structures to handle both queries $AllPairs(t, d)$ and $Beginnings(t, d)$. In Section 5, we discuss some theoretical and experimental results related to the number of special pairs in a time series and we also describe the experiments executed to evaluate the peformance of our solutions. Finally, in Section 6, we draw our conclusion and discuss some possible extensions.

2 Basic Concepts

In this section we present observations/results that are useful to handle both queries $AllPairs(t, d)$ and $Beginning(t, d)$.

Our first observation is that it is easy to find all (t, d)-events that have a given time index i as a starting point. For that, we need a data structure that supports range maximum queries over the input A in constant time and with linear preprocessing time/space, as discussed in the related work section. Having this data structure in hands, we call the procedure `GenEventStart`, presented in Figure 1, with parameters $(i, i+1, i+t)$. This procedure, when executed with parameters $(i, low, high)$, generates all (t, d)-events with startpoint i and with endpoint in the range $[low, high]$. First the procedure uses the data structure to find the time index j with maximum value in the range $[low, high]$ of vector A. If the deviation of (i, j) is smaller than d then the search is aborted because there are no (t, d)-events that satisfy the required conditions. Otherwise, it reports (i, j) and recurses on intervals $[low, j-1]$ and $[j+1, high]$.

The running time of the procedure is proportional to the number of recursive calls it executes. The execution flow can be seen as a binary tree where each node corresponds to a recursive call. At each internal node a new (t, d)-event is generated. Since the number of nodes in a binary tree is at most twice the number of internal nodes, it follows that the number of recursive calls is at most $2k_i$, where k_i is the number of (t, d)-events that have startpoint i and endpoint in the range $[low, high]$. Our discussion is summarized in the following proposition.

Proposition 1. *Let k_i be the number of (t, d)-events that have startpoint i and endpoint in the range $[low, high]$. Then, the procedure* `GenEventsStart` *(i,low,high) generates all these (t, d)-events in $O(k_i + 1)$ time.*

We shall consider the existence of an analogous procedure, which we call `GenEventsEnd`, that receives as input a triple $(j, low, high)$ and generates all (t, d) events that have j as an endpoint and startpoints in the range $(low, high)$.

The following lemma motivates the focus on the special pairs in order to handle queries $AllPairs(t, d)$ and $Beginning(t, d)$.

Lemma 1. *Let (i, j) be a (t, d)-event. Then, the following conditions hold*

```
GenEventsStart($i$,$low$,$high$)
    If $high \geq low$
        $j \leftarrow RMQ(low, high)$
        If $a_j - a_i \geq d$   (*)
            Add $(i, j)$ to the list of $(t, d)$-events   (**)
            GenEventsStart($i$,$low$,$j$-1)
            GenEventsStart($i$, $j$+1,$high$)
        End If
    End If
```

Fig. 1. Procedure to generate (t, d)-events that have a given startpoint

i there is an index i^* such that (i^*, j) is a (t, d)-event and i^* is the startpoint of a special (t, d)-event.

ii there is an index j^* such that (i, j^*) is a (t, d)-event and j^* is the endpoint of a special (t, d)-event.

Proof. We just need to prove (i) because the proof for (ii) is analogous. Let i^* be the time index of the element of A with minimum a_{i^*} among those with time indexes in the set $\{i, \ldots, j-1\}$. In case of ties, we consider the largest index. Because $a_{i^*} \leq a_i$, $i \leq i^* < j$ and (i, j) is a (t, d)-event, we have that (i^*, j) is also a (t, d)-event. Let k^* be the time index of the element of A with maximum value among those with time indexes in the set $\{i^* + 1, \ldots, j\}$. The pair (i^*, k^*) is both a special pair and a (t, d)-event, which establishes (i). □

Our last result in this section shows that the set S of special pairs of a given time series of size n can be generated in $O(|S| + n)$ time. This is accomplished by calling the pseudo-code presented in Figure 2 with parameters $(1, n)$. First, the procedure uses a RMQ data structure to compute in constant time the time index i with the smallest value in the range $[low, high]$. At this point, it concludes that there are no special pairs (x, y) with $x < i < y$ because $a_x > a_i$. Then, it generates all special pairs with startpoint i by invoking a modified version of `GenEventsStart`. Next, it looks for, recursively, for special pairs with time indexes in the range $(low, i-1)$ and also for those with time indexes in the range $(i + 1, high)$.

The procedure `ModifiedGenEventsStart` is similar to `GenEventsStart` but for the following differences: it verifies whether $a_j > a_i$ in line (*) and it adds the pair to the list of special pairs in line (**) of the pseudo-code of Figure 1

This last result is summarized in the following proposition.

Proposition 2. *Let S be the set of special pairs of a time series A of size n. Then, S can be constructed in $O(|S| + n)$ time*

3 The Query AllPairs(t, d)

We propose two algorithms to process query *AllPairs*(t, d), namely, AllPairs-RMQ and AllPairs-SP. The first one relies on a data structure of size $O(n)$ and

```
GenSpecialPairs($low$,$high$)
    If $high \geq low$
        $i \leftarrow RMinQ(low, high)$
        ModifiedGenEventsStart($i, i+1, high$)
        GenSpecialPairs($low, i-1$)
        GenSpecialPairs($i+1, high$)
    End If
```

Fig. 2. Procedure to generate all special pairs

process $AllPairs(t, d)$ in $O(n + k)$ time, where k is the size of the solution set. The second one requires $O(n+|S|)$ space and process $AllPairs(t, d)$ in $O(k)$ time, where $|S|$ is the number of special pairs of the time series under consideration.

3.1 Algorithm AllPairs-RMQ

AllPairs-RMQ builds, during its preprocessing phase, a data structure to support range maximum queries over A. The algorithms spends $O(n)$ time in this phase to build a structure of $O(n)$ size.

To process $AllPairs(t, d)$, it scans the list A and for each time index i, it calls `GenEventsStart`$(i, i+1, \min\{i+t, n\})$. It follows from Proposition 1 that it process $AllPairs(t, d)$ in $O(n+k)$ time, where k is the number of (t, d)-events in the solution set.

3.2 Algorithm AllPairs-SP

Preprocessing Phase. Let S denote the set of special pairs of the time series $A = (a_1, ..., a_n)$. In it preprocessing phase, AllPairs-SP generates the set S and stores its pairs, sorted by increasing order of its Δ-values, in a vector V; ties are arbitrarily broken. Furthermore, it builds an auxiliary RMQ data structure $\mathcal{D}$ of size $O(|S|)$ to be able to answer the following query: given two values $t^\star$ and $d^\star$, report the subset of special pairs with Δ-value at most $t^\star$ and deviation at least $d^\star$. Finally, it builds two auxiliary RMQ data structures $\mathcal{D}_{min}$ and $\mathcal{D}_{max}$ of size $O(n)$ to support, respectively, `GenEventsEnd` and `GenEventsStart` procedure calls over A.

To handle the query $AllPairs(t, d)$, we execute two phases. In the first one, we retrieve all special (t, d)-events from V by using the data structure $\mathcal{D}$. Then, in the second phase, we use these special (t, d)-events and the structures $\mathcal{D}_{min}$ and $\mathcal{D}_{max}$ to retrieve the other (t, d)-events. The two phases are detailed below.

Phase 1. Let k^* be the time index of the last pair in V among those with Δ-value at most t.[1] We use $\mathcal{D}$ to perform a set of maximum range queries to find the (t, d)-events in $V[1, .., k^*]$. More precisely, let i be the index of V returned by a range maximum query over V with input range $[1, k^*]$. If the deviation of

[1] It must be observed that k^* can be found in $O(1)$ time, for a given t, by preprocessing V with $O(n)$ time/space consumption.

the pair stored in $V[i]$ is smaller than d we abort the search because no other pair in $V[1,..,k^*]$ has deviation larger than d. Otherwise, we add this pair to a list $\mathcal{L}$ and we recurse on subvectors $V[1,..,i-1]$ and $V[i+1,..,k^*]$. At the end of this phase the list $\mathcal{L}$ contains all special (t,d)-events.

Phase 2. This phase can be split into two subphases:

Phase 2.1. In this subphase, we generate the set of distinct startpoints of all (t,d)-events. First, we scan the list $\mathcal{L}$ to obtain a list $E_{\mathcal{L}}$ containing the distinct endpoints of the special (t,d)-events in $\mathcal{L}$. This can be accomplished in $O(|\mathcal{L}|)$ time by keeping an $0-1$ vector to avoid repetitions among endpoints. Then, we invoke the procedure `GenEventsEnd`$(j, j-t, j-1)$, which is supported by the data structure $\mathcal{D}_{min}$, for every $j \in E_{\mathcal{L}}$. We keep track of the startpoints of the (t,d)-events generated in this process by storing them in a list B. Again, we use a $0-1$ vector to avoid repetitions among startpoints. It follows from item (ii) of Lemma 1 that list B contains the startpoints of all (t,d)-events.

Phase 2.2. In this subphase, we generate all (t,d)-events that have startpoints in B. For that, we invoke `GenEventsStart`$(i, i+1, i+t)$ for every every $i \in B$. One must recall that `GenEventsStart` is supported by the data structure $\mathcal{D}_{max}$. The list of pairs of indexes (i,j) returned by `GenEventsStart` will be the (t,d)-events we were looking for.

The results of this section can be summarized in the following theorem.

Theorem 1. *The algorithm AllPairs-SP generates all the k solutions of the query $AllPairs(t,d)$ in $O(1+k)$ time and with $O(|S|+n)$ preprocessing time/space.*

4 The Query Beginning(t,d)

We propose three algorithms that differ in the time/space required to process $Beginning(t,d)$.

4.1 Beg-Quick

This algorithm scans the time series A checking, for each time index i, whether i is the startpoint of a (t,d)-event. To accomplish that it makes use of a list Q of size $O(t)$ that stores some specific time indexes. We say that a time index j is dominated by a time index k if and only if $k > j$ and $a_k \geq a_j$. The algorithm mantains the following invariants: right before testing whether $i-1$ is a startpoint of a (t,d)-event, the list Q stores all time indexes in the set $\{i,\ldots,(i-1)+t\}$ that are not dominated by other time indexes from this same set. Moreover, Q is simultaneously sorted by increasing order of time indexes and non-increasing order of values.

Due to these invariants, the time index of the largest a_j, with $j \in \{i,\ldots,(i-1)+t\}$ is stored at the head of Q so that we can decide whether $i-1$ belongs to the solution set by testing if $a_{Head(Q)} - a_{i-1}$ is at least d. In the positive case, $i-1$ is added to the solution set. To guarantee the maintainance of these invariants, for the next loop, the algorithm removes i from the head of Q (if it is still there); it removes all time indexes dominated by $i+t$ and, finally, it adds $i+t$ to the tail of Q.

The algorithm runs in $O(n)$ time. To see that note that each time index is added/removed to/from Q exactly once. The pseudo-code for the algorithm is presented in Figure 3. The first block is used to intitialize the list Q so that it respects the above mentioned invariants, right before testing whether 1 belongs to the solution set.

```
Add time index $t+1$ to $Q$
**For** $k = t, ..., 2$
        If $a_k$ is larger than $a_{Head(Q)}$ **then** Add $k$ to the head of $Q$
**End For**
**If** $(1, head(Q))$ is a $(t,d)$-event **then** Add time index 1 to the solution set
**For** $i = 2, .., n-1$
        **If** $head(Q) = i$, remove $i$ from the head of $Q$
        **If** $i + t \leq n$
            Traverse $Q$ from its tail to its head removing every time index $j$ that is
            dominated by $i+t$
            Add $(i+t)$ to the tail of Q
        **End If**
        **If** $(i, head(Q))$ is a $(t,d)$-event **then** Add $i$ to the solution set
**End For**
```

Fig. 3. The Procedure Beg-Quick

4.2 Algorithm Beg-SP

Preprocessing Phase. First, the set S of special pairs is generated. Then, S is sorted by decreasing order of deviations. Next, we build an ordered binary tree $\mathcal{T}$ where each of its nodes is associated with a subinterval of $[1, n-1]$. More precisely, the root of $\mathcal{T}$ is associated with the interval $[1, n-1]$. If a node v is associated with an interval $[i, .., j]$ then the left child of v is associated with $[i, \lfloor (i+j)/2 \rfloor]$ and its right child is associated with $[\lfloor (i+j)/2 \rfloor + 1, j]$. Furthermore, each node of $\mathcal{T}$ points to a list associated with its interval, that is, if v is associated with an interval $[i, j]$ then v points to a list $L_{i,j}$. Initially, all these lists are empty.

Then, the set S is scanned in non-increasing order of deviations of its special pairs. If a special pair s has Δ-value t then s is appended to all lists $L_{i,j}$ such that $i \leq t \leq j$. By the end of this process each list $L_{i,j}$ is sorted by non-increasing order of the deviations of its special pairs. Finally, we scan each list $L_{i,j}$ and remove a special pair whenever its endpoint has already appeared as an endpoint of another special pair.

The size of this structure is $O(\min\{n^2, |S| \log n\})$ because each special pair may occur in at most $\log n$ lists $L_{i,j}$ and the size of each of these n lists is upper bounded by n because for each endpoint at most one special pair is stored.

To handle a query $Beginning(t, d)$ we execute two phases. In the first one, we retrieve the endpoints of the special (t, d)-events. In the second phase, we use these endpoints to retrieve the startpoints of the (t, d)-events.

The correctness of this approach relies on item (ii) of Lemma 1 because if i is a startpoint of a (t, d)-event then there is an endpoint j^* of a special (t, d)-event such that (i, j^*) is a (t, d)-event. As a consequence, the endpoint j^* is found at Phase 1 and it is used to reach i at Phase 2. In the sequel, we detail these phases

Phase 1. First, we find a set K of nodes in $\mathcal{T}$ whose associated intervals form a partition of the interval $[1, t]$. If $n = 8$ and $t = 6$, as an example, K consists of two nodes: one associated with the interval $[1, 4]$ and the other with the interval $[5, 6]$. It can be shown that it is always possible to find in $O(\log n)$ time a set K with at most $\log t$ nodes. One way to see that is to realize that tree $\mathcal{T}$ is exactly the first layer of a 2D range search tree (see e.g. Chapter 5 of [4]).

Then, for each node $v \in K$ we proceed as follows: we scan the list of special pairs associated with v; if the current special pair in our scan has deviation larger than d, we verify whether it has already been added to the list of endpoints E_f. In the negative case we add it to E_f; otherwise, we discard it. The scan is aborted when a special pair with deviation smaller than d is found. This step spends $O(|E_f| \log t)$ because each endpoint may appear in at most $\log t$ lists.

Phase 2. The second phase can be split into three subphases:

Phase 2.1. In the first subphase, we sort the endpoints of E_f in $O(\min\{n, |E_f| \log |E_f|\})$ time by using either a bucket sort or a heapsort algorithm depending whether $|E_f|$ is larger than $n/\log n$ or not. Let $e_1 < e_2 < \ldots < e_f$ be the endpoints in E_f after applying the sorting procedure.

Phase 2.2. To generate the beginnings efficiently, it will be useful to calculate the closest larger predecessor (*clp*) of each endpoint $e_j \in E_f$. For $j = 1, \ldots, |E_f|$, we define $clp(e_j) = e_{i^*}$, where $i^* = \max\{i | e_i \in E_f \text{ and } e_i < e_j \text{ and } a_{e_i} \geq a_{e_j}\}$. To guarantee that the *clp*'s are well defined we assume that E_f contains an artificial endpoint e_0 such that $e_0 = 0$ and $a_{e_0} = \infty$. As an example, let $E_5 = \{e_1, e_2, e_3, e_4, e_5\}$ be a list of sorted endpoints, where $(e_1, e_2, e_3, e_4, e_5) = (2, 3, 5, 8, 11)$ and $(a_{e_1}, a_{e_2}, a_{e_3}, a_{e_4}, a_{e_5}) = (8, 5, 7, 6, 4)$. We have that $clp(e_4) = e_3$ and $clp(e_3) = e_1$. The *clp*'s can be calculated in $O(|E_f|)$ time.

Phase 2.3. To generate the beginnings of the (t, d)-events from the endpoints in E_f, the procedure presented in Figure 4 is executed. The procedure iterates over all endpoints in E_f and it stores in the variable $CurrentHigh$ the smallest time index j for which all beginnings larger than j have already been generated. This variable is initialized with value $e_f - 1$ because there are no beginnings larger that $e_f - 1$. For each endpoint e, the procedure looks for new startpoints in the range $[\max\{e_i - t, clp(e_i)\}, CurrentHigh]$. It does not look for startpoints in the range $[e_i - t, clp(e_i)]$ because every startpoint in this range that can be generated from e_i can be also generated from $clp(e_i)$ while the opposite is not necessarily true. Indeed, this is the reason why we calculate the *clp*'s – they avoid to generate a startpoint more than once. Next, the variable $CurrrentHigh$ is updated and a new iteration starts. This subphase can be implemented in $O(|E_f| + k)$ time, where k is the number of beginnings generated.

Our discussion is summarized in the following theorem.

Theorem 2. *The algorithm Beg-SP generates all the k solutions of the query Beginning(t, d) in $O(\log n + f \cdot (\log f + \log t) + k)$ time, where f is the number of distinct endpoints of special (t, d)-events. Furthermore, the algorithm employs a data structure of size $O(\min\{n^2, |S| \log |S| + n\})$ that is built in $O(|S| \log |S| + n)$ time in the preprocessing phase.*

As we have already mentioned, we expect f to be smaller than k due to the symmetry between startpoints and endpoints, and the fact that f is the number of endpoints for a restricted class of (t, d)-events(the special ones) while k is the number of startpoints of unrestricted (t, d)-events. In fact, we have observed this behavior, $f < k$, for more than thousand queries executed over real time series as described in Section 5.2.

```
CurrentHigh ← $e_f - 1$
For $i = f$ downto 1.
        GenEventsEnd  $(e_i, \max\{e_i - t, clp(e_i)\}, CurrentHigh)$
        CurrentHigh ← $\max\{e_i - t, clp(e_i)\}$
End For
```

Fig. 4. Procedure to generate the beginnings

4.3 Beg-Hybrid

Another way to process query *Beginning*(t, d) is to combine the algorithms AllPairs-SP and Beg-SP as follows:

1. Execute the preprocessing phase of AllPairs-SP, skipping the construction of data structure $\mathcal{D}_{max}$ since it is not useful for processing *Beginning*(t, d).
2. Apply Phase 1 of AllPairs-SP. At the end of this phase we have a list $\mathcal{L}$ containing all the special (t, d) events.
3. Scan the list $\mathcal{L}$ to obtain the list E_f containing the distinct endpoints of the specials (t, d)-events. Apply Phase 2 of Beg-SP.

The main motivation of this approach, when contrasted with Beg-SP, is the economy of a $\log n$ factor in the space consumption of the underlying data structure. Another motivation is the fact that it uses the same date structure employed by AllPairs-SP so that both *AllPairs*(t, d) and *Beginning*(t, d) can be processed with a single data structure. Its disadvantage, however, is that it requires $O(k \min\{t, f\})$ time, rather than $(\log n + f \cdot (\log f + \log t) + k)$ time, to process *Beginning*(t, d), where k is the size of the solution set and f is the number of distinct endpoints of special (t, d)-events.

The correctness of Beg-Hybrid follows directly from the correctness of both AllPairs-SP and Beg-SP.

5 Experimental Work

5.1 On the Number of Special Pairs

Since the preprocessing time and space of some of our data structures depend on the number of special pairs, we dedicate this section to discuss some theoretical and practical results related to this quantity.

Clearly, the number of special pairs of a time series is at least 0 and at most $n(n-1)/2$, where the lower(upper) bound is reached for a decreasing(increasing) series. The following proposition shows that the expected number of special pairs for a random permutation of the n first integers is $n - H_n$. This result is interesting in the sense that the number of special pairs of a time series, with distinct values, is equal to the number of special pairs of the permutation of $\{1, \dots, n\}$ in which the time series can be naturally mapped on.

Proposition 3. *Let S be a list of special pairs constructed from a time series taken uniformly at random from a set of* n *elements. Then, the expected size of S is $n - H_n$, where H_n is the* n*-th harmonic number.*

Proof. Let $E[X]$ represent the expected size of list S of special pairs. Furthermore, let $X_{i,j}$ denote a random indicator variable that stores 1 if the pair of time indexes (i, j) belongs to S and 0 otherwise.

From the previous definitions, $E[X] = E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}]$.

By the linearity of expectation, it follows that:

$$E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{i,j}].$$

Since $E[X_{i,j}] = \frac{1}{(j-i+1)(j-i)} = \frac{1}{j-i} - \frac{1}{j-i+1}$, we have:

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{1}{j-i} - \frac{1}{j-i+1} \right) = \sum_{i=1}^{n-1} \left(1 - \frac{1}{n-i+1} \right) = n - \sum_{i=1}^{n} \frac{1}{i} = n - H_n.$$

We shall mention that it is possible to show that the cardinality of S is highly concentrated around its mean value.

Since our research is strongly motivated by applications that arise in finance domain, we have also evaluated the number of special pairs for time series containing the prices of 48 different Brazilian stocks, sampled in a minute basis, over a period of three years. We have also considered the number of special pairs for the inverted time series. These inverted series shall be used if one is looking for negative variations rather than positive ones. We measured the ratio between the number of special pairs and the size of the underlying time series for each of these 96 series. We observe that this ratio lies in the range $[0.5, 7]$, with median value 2.50, and average value 2.75.

The results of this section suggest that the data structures that depend on the number of special pairs shall have a very reasonable space consumption for practical applications. We shall note that we can always set an upper bound on the number of special pairs to be stored and recourse to a structure that do not use special pairs if this upper bound is reached.

5.2 On the Preprocessing/Querying Elapsed Time

In this section we present the experiments that we carried on to evaluate the performance of the proposed data structures.

For these experiments, we selected, among the 96 time series available, those with the largest size, which accounts to 38 times series, all of them with 229875 samples. Our codes were developed in `C++`, using compiler `o g++ 4.4.3`, with the optimization flag `-O2` activated. They were executed in a 64 bits Intel(R) Core(TM)2 Duo CPU, T6600 @ 2.20GHz, with `3GB` of RAM.

To support RMQ's we implemented one of the data structure that obtained the best results in the experimental study described in [6]. This data structure requires linear time/space preprocessing and handles RMQ's over the range $[i, j]$ in $O(\min\{\log n, j - i\})$ time.

The following table presents some statistics about the time(ms) spent in the preprocessing phase of our algorithms.

Algorithm	**Min Time**	**Average Time**	**Max Time**
AllPairs-RMQ	7.8	8.7	10.1
Beg-Hybrid	44	128	302
Beg-SP	165	681	1601
Beg-Quick	N/A	N/A	N/A

Experiments with Query Beginning(t, d). In our experiments, we also included a naive method, denoted by Beg-Naive, that for each time index i, it looks for the first time index j in the interval $[i + 1, i + t]$ such that $a_j - a_i \geq d$. Clearly, this method spends $O(nt)$ to process $Beginning(t, d)$.

Since we are interested in rare events, we focused on pairs (t, d) that induce a small number of solutions for $Beginning(t, d)$, that is, the size of the output is a small percentage of the input's size. For each of the 38 series, we executed 30 queries, with $1 \leq t \leq 61$.

The elapsed times are plotted in Figure 5. The horizontal axis shows the ratio (in percentage) between the size of the output and the size of the input series while the vertical axis shows the query elapsed time.

First, we observe that Beg-Naive is hugely outperformed by the other methods. The dispersion in the results of Beg-Quick is due to the dependence of its query elapsed time with the value of t. We observe also that algorithm Beg-SP outperforms Beg-Quick when the size of the output is much smaller than the input's size. As this ratio gets larger, the difference becomes less significant. In general, when the size of the output is $p\%$ of the input's size, Beg-SP is $10/p$ times faster, in average, than Beg-Quick. As an example, when $p = 0.1\%$, Beg-SP is 100 times faster.

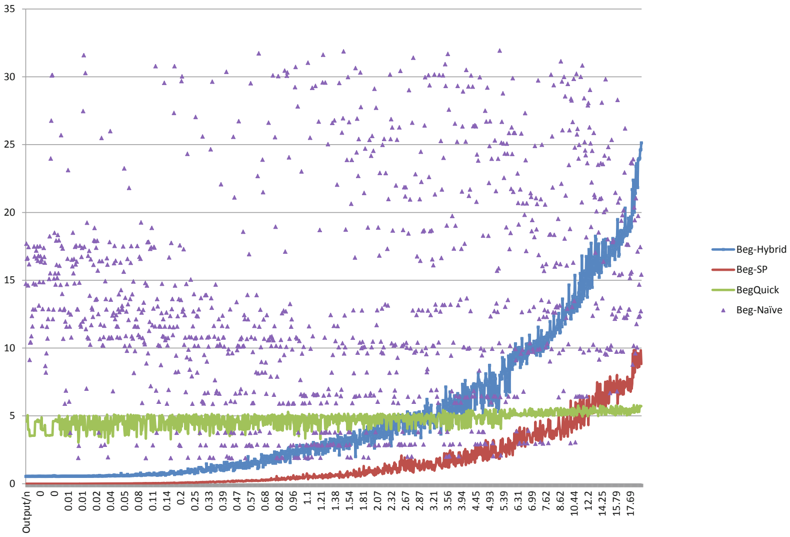


Fig. 5. Elapsed query times of the algorithms for processing Beginning(t, d))

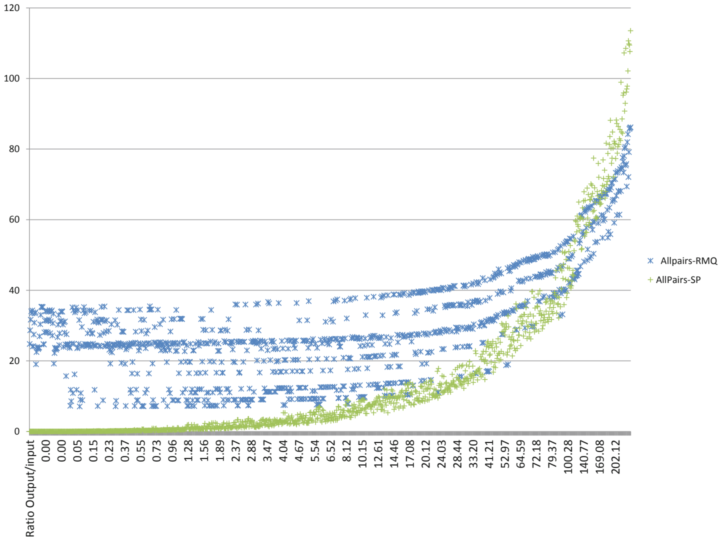


Fig. 6. Elapsed query times of AllPairs-SP and AllPairs-RMQ

It is also interesting to observe that Beg-Hybrid, though slower than Beg-SP, also yields a reasonable gain when compared with Beg-Quick to detect very rare variations.

Recall that Theorem 2 states that Beg-SP spends $O(\log n + f \cdot (\log f + \log t) + k)$ time to process $Beginning(t, d)$, where k is the size of the solution set and f is the number of distinct endpoints of special (t, d)-events. Hence, it is interesting to understand the relation between f and k. We observed that this ratio is smaller than 1 for all queries that produced at least 10 outputs. Thus, one should expect to pay $O(\log k + \log t)$ time per output generated by algorithm Beg-SP.

Experiments with Query AllPairs(t, d). We executed 30 queries AllPairs(t, d) for each of the 38 series with 229875 samples. The elapsed times are plotted in Figure 6. As expected, AllPairs-SP outperforms AllPairs-RMQ when the size of the output is much smaller than the size of the time series. We have not performed other experiments due to the lack of space and because we understand that the theoretical analysis of AllPairs-SP and AllPairs-RMQ succeed in explaining their behavior.

6 Conclusions

In this paper, we proposed and analyzed algorithms/data structures to detect rare variations in time series. Our solutions combine data structures that are well known in the computational geometry community, as those that support RMQ's, with the notion of a special pair. We present some theoretical results and carried on a set of experiments that suggest that our solutions are very efficient in terms of query elapsed time and are compact enough to be used in practical situations.

Although we focused on two particular queries, $Beginning(t, d)$ and $AllPairs$ (t, d), the approach presented here can be extended to other queries with the same flavor. As a future work, we aim to investigate how to extend our techniques to deal efficiently with multiple time series and with queries that should count the number of solutions rather than reporting them.

References

1. Bender, M.A., Farach-colton, M.: The lca problem revisited. In: In Latin American Theoretical INformatics, pp. 88–94. Springer (2000)
2. Bentley, J.L., Stanat, D.F., Hollings Williams Jr., E.: The complexity of finding fixed-radius near neighbors. Inf. Process. Lett. 6(6), 209–212 (1977)
3. Chaudhuri, K., Wu, Y.: Mean reversion in stock prices: evidence from emerging markets. Managerial Finance 29(10), 22–37 (2003)
4. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. Springer (January 2000)
5. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Proceedings of ACM SIGMOD, Minneapolis, MN, pp. 419–429 (1994)
6. Fischer, J., Heun, V.: Theoretical and Practical Improvements on the RMQ-Problem, with Applications to LCA and LCE. In: Lewenstein, M., Valiente, G. (eds.) CPM 2006. LNCS, vol. 4009, pp. 36–48. Springer, Heidelberg (2006)

7. Fu, T.: A review on time series data mining. Engineering Applications of Artificial Intelligence (24), 164–181 (2011)
8. Gatev, E., Goetzman, W.N., Rouwenhorst, K.G.: Pairs trading: Performance of a relative-value arbitrage rule. Review of Financial Studies 3(19), 797–827 (2007)
9. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. 13, 338–355 (1984)
10. Lim, S.-H., Park, H., Kim, S.-W.: Using multiple indexes for efficient subsequence matching in time-series databases. Inf. Sci 177(24), 5691–5706 (2007)
11. Lo, A., Mamaysky, H., Wang, J.: Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. Journal of Finance 2(1), 1705–1770 (2000)
12. Moon, Y.-S., Whang, K.-Y., Han, W.-S.: General match: a subsequence matching method in time-series databases based on generalized windows. In: Franklin, M., Moon, B., Ailamaki, A. (eds.) Proceedings of the ACM SIGMOD International Conference on Management of Data, Madison, WI, USA, June 3-6, pp. 382–393. ACM Press (2002)
13. Shasha, D., Zhu, Y.: High performance discovery in time series: techniques and case studies. Springer (2004)
14. Wu, H., Salzberg, B., Zhang, D.: Online event-driven subsequence matching over financial data streams. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data 2004, Paris, France, June 13-18, pp. 23–34. ACM Press (2004)

Invariant Time-Series Classification

Josif Grabocka[1], Alexandros Nanopoulos[2], and Lars Schmidt-Thieme[1]

[1] Information Systems and Machine Learning Lab
Samelsonplatz 22, 31141 Hildesheim, Germany
[2] University of Eichstaett-Ingolstadt, Germany
{josif,schmidt-thieme}@ismll.de, alexandros.nanopoulos@ku.de

Abstract. Time-series classification is a field of machine learning that has attracted considerable focus during the recent decades. The large number of time-series application areas ranges from medical diagnosis up to financial econometrics. Support Vector Machines (SVMs) are reported to perform non-optimally in the domain of time series, because they suffer detecting similarities in the lack of abundant training instances. In this study we present a novel time-series transformation method which significantly improves the performance of SVMs. Our novel transformation method is used to enlarge the training set through creating new transformed instances from the support vector instances. The new transformed instances encapsulate the necessary intra-class variations required to redefine the maximum margin decision boundary. The proposed transformation method utilizes the variance distributions from the intra-class warping maps to build transformation fields, which are applied to series instances using the Moving Least Squares algorithm. Extensive experimentations on 35 time series datasets demonstrate the superiority of the proposed method compared to both the Dynamic Time Warping version of the Nearest Neighbor and the SVMs classifiers, outperforming them in the majority of the experiments.

Keywords: Machine Learning, Time Series Classification, Data-Driven Transformations, Invariant Classification.

1 Introduction

Time series classification is one of the most appealing research domains in machine learning. The generality of interest is influenced by the large number of problems involving time series, ranging from financial econometrics up to medical diagnosis [1]. The most widely applied time series classifier is the nearest neighbor (NN) empowered with a similarity/distance metric called Dynamic Time Warping (DTW), hereafter jointly denoted as DTW-NN [2]. Due to the accuracy of DTW in detecting pattern variations, DTW-NN has been characterized as a hard-to-beat baseline [3].

Support Vector Machines (SVMs) are successful classifiers involved in solving a variety of learning and function estimation problems. Yet, experimental studies show that it performs non-optimally in the domain of time series [4]. We explore

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 725–740, 2012.

an approach to boost the classification of time series using SVMs, which is directly inspired by the nature of the problem and the reasons why SVMs fail to build optimal decision boundaries. In most time series datasets, the variations of instances belonging to the same class, denoted intra-class variation, are considerably numerous. Variations appear in different flavors. A pattern of a signal/time series can start at various time points (translational variance) and the duration of a pattern can vary in length (scaling variance). Even more challenging, such variances can partially occur in a signal, in multiple locations, by unexpected direction and magnitude of change. There exist more, theoretically infinitely many, possible variations of a particular class pattern, compared to the present number of instances in the dataset. Ergo, such lack of sufficient instances to cover all the possible variations can affect the maximum margin decision boundary in under-representing the ideal decision boundary of the problem.

In order to overcome the lack of instances, the insertion of virtual transformed instances to the training set has been proposed. In the case of SVMs, support vectors are transformed/deformed, the new virtual support vectors added back to the training set and the model is finally retrained [5,6]. An illustration of the effect of inserting virtual instances and its impact on redefining the decision boundary is shown in Figure 1. The most challenging aspect of this strategy is to define efficient transformation functions, which create new instances from existing ones, enabling the generated instances to represent the necessary variations in the feature space.

The main contribution of our study is in defining a novel instance transformation method which improves the performance of SVMs in time series classification. In our analysis the transformations should possess four characteristics. First the transformations should be data-driven, concretely an analysis of the intra-class variance distributions should be taken into account. Secondly, localized variations are required since the variations of series instances appears in forms of local deformations. Thirdly, in order to overcome the time complexity issues, only a representative subset of the instances should be selected for producing variations. Finally the transformations should accurately redefine the decision boundary without creating outliers or over-fit the training set.

The novel transformation method we introduce satisfies all the above raised requirements. The proposed method analyses the translational variance distributions by constructing warping alignment maps of intra-class instances. The time series is divided into a number of local regions and transformation fields/vectors are created to represent the direction and magnitude of the translational variance at every region center, based on the constructed variance distributions. Finally, the application of the transformation fields to time series is conducted using the Moving Least Squares algorithm [7].

The efficiency of the proposed method is verified through extensive experimentation on 35 datasets from the UCR collection[1]. Our method clearly outperforms DTW-NN on the vast majority of challenging datasets, while being on a par competitive with DTW-NN in the easy (low error) ones. Furthermore, the

[1] www.cs.ucr.edu/~eamonn/time_series_data

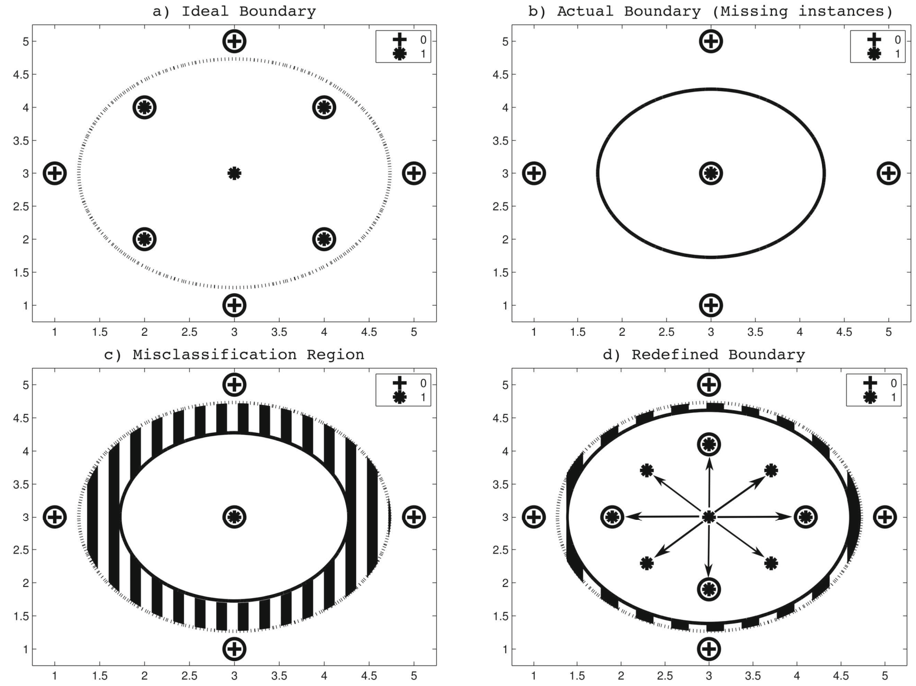


Fig. 1. a) An ideal max-margin decision boundary for the depicted binary (0/1) classification problem. *Circled* points denote support vectors. **b)** An actual version of the problem where most class 1 instances are missing. The actual max-margin decision boundary differs from the ideal boundary in a). **c)** The under-fitting of the actual boundary (*solid*) to represent the ideal boundary (*dots*) produces the *shaded* misclassification region. **d)** Transforming the class 1 instance in coordinate (3,3) and inserting the transformed instances (*pointed by arrows*) back to the dataset, helps redefine the new max-margin boundary (*solid*). Consecutively, the area of the misclassification region is reduced.

results indicate that our proposed method always improves the default SVM. The principal contributions of the study can be summarized as:

- A novel time series transformation method is presented
- For the first time, the approach of Invariant SVMs is proposed in time-series domain
- Extensive experimentations are conducted to demonstrate the superiority of the proposed method

2 Related Work

2.1 Time Series Classification

Time series classification has been elaborated for more than two decades and a plethora of methods has been proposed for the task. Various classifiers have been applied, ranging from neural networks [8,9] to bayesian networks [10], from decision trees [11] to first-order logic rules [12], and from Hidden Markov Models [13] to tailored dimensionality reduction [14].

Dynamic Time Warping. Despite the major effort spent in building accurate time series classifiers, still the nearest neighbor classifier combined with a similarity technique called Dynamic Time Warping (DTW) was reported to produce more accurate results [15]. DTW overcomes the drawback of other methods because it can detect pattern variations, such as translations/shifts, size and deformations. The metric builds an optimal alignment of points among two time series by dynamically constructing a progressive cost matrix. It computes the the path of the minimal overall point pairs' distance [2]. Adding warping window size constraints have been reported to occasionally boost the classification [16]. In contrast to DTW-NN, our study aims at building a competitive max-margin classifier.

2.2 Invariant SVMs Classification

Even though the challenge of handling invariance in time series classification is rather new, it has, however, been applied long ago to the domain of image classification. Significant effort to the problem of invariant classification was followed by the SVM community, where one of the initial and most successful approaches relies on creating virtual instances by replicating instances. Typically the support vectors are transformed creating new instances called Virtual Support Vectors (VSV), with the aim of redefining the decision boundary [5,17]. VSV has been reported to achieve optimal performance in image classification [6]. An alternative technique in handling variations relies in modifying the kernel of the SVM, by adding a loss term in the dual objective function. Such a loss enforces the decision boundary to be tangent to the transformation vector [6,18]. Other approaches have been focused on selecting an adequate set of instances for transformation [19]. Studies aiming the adoption of SVM to the context of time series have been primarily addressing the inclusion of DTW as a kernel [20,21]. Unfortunately, to date, all proposed DTW-based kernels we are aware of, are not efficiently obeying the positively semi-definite requirement [4]. The DTW based kernels have been reported to not perform optimally compared to state-of-art [22]. Generating new instances based on the pairwise similarities has also been applied [4], with limited success compared to DTW-NN. In comparison, our method applies a novel transformation technique along with the VSV approach.

2.3 Instance Transformations

Intentional transformations and deformations of time series has shown little interest because of the limited studies of VSV classifiers in the domain. Among the initiatives, morphing transformation from a time series to another has been inspected [23]. However, deformations have been more principally investigated in the domain of images. Moving Least Squares is a state-of-art technique to produce realistic deformations [7]. In this work we use the Moving Least Squares method for applying the transformations over series.

3 Proposed Method

3.1 Principle

In order to boost the classification accuracy our method needs to generate new instances via transformations. In order to capture the necessary patterns' intra-class variations, the transformation technique should aim for certain characteristics and requirements. In our analysis, the transformations should obey to the following list of properties:

- **Data-Driven:** Variance should be generated by analyzing the similarity distribution of instances inside a class.
- **Localized:** Intra-class variations are often expressed in local deformations, instead of global variations.
- **Selective:** Transforming all the instances becomes computationally expensive and many instances can be redundant w.r.t to the decision boundary. Therefore it is crucial to select only a few class-representative instances for generating variations.
- **Accurate:** The transformed instances should help redefine the decision boundary, however care should be payed to avoid excessive magnitudes of transformation, in order to avoid generating outliers.

3.2 Method Outline

The transformation method and the instance generation technique we are introducing, does answer all the requirements we raised in section 3.1. Initially we define the local transformation fields in subsection 3.3, which are used to transform time series using the Moving Least Squares algorithm. The transformation fields are constructed by measuring the translational variance distributions subsection 3.5. The variance distributions are obtained by building the intra-class warping maps, defined in subsection 3.4. Finally, the transformation fields are applied only to the support vectors following the Virtual Support Vector classification approach, defined in subsection 3.6.

3.3 Transformation Fields and Moving Least Squares

In this subsection we present only the technicalities of how a time-series can be transformed by a particular localized transformation, while the actual method that creates intelligent transformation magnitudes will be introduced in forthcoming subsections. Variations of time series are often occurring in localized forms, meaning that two series differ only in the deformation of a particular subsequence rather than a global difference. In order to include the mechanism of localized variances we first introduce the concept of a **transformation field**, denoted F. We split the time series into K many regions, and then we define the left/right translation that is required for each region. Each region will be transformed dedicatedly, while the transformation will be applied to the centroid of the region. Such centroids are denoted as **control points**. The amount of translational transformation applied to every control point (hence every region) is denoted as the transformation field vector $F \in \mathbb{R}^K$. For instance Figure 2 shows the effect of applying a transformation field on two regions of a time series, where each region is denoted by its representative control point.

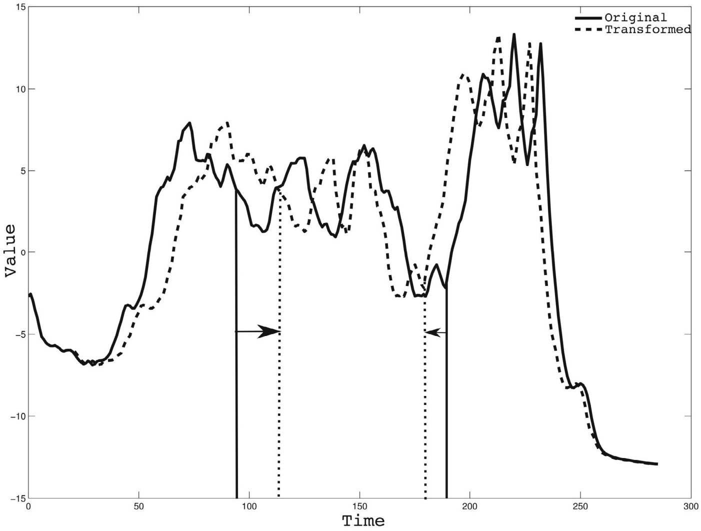


Fig. 2. Demonstrating the effect of applying a transformation field vector of values [+20 -10] to the control points positioned at [95 190] highlighted with vertical lines, on an instance series from the *Coffee* dataset. The transformation algorithm (MLS) is described in Algorithm 1.

The mechanism of applying a transformation field to a series is conducted via the deformation algorithm called Moving Least Squares (MLS), which is described in Algorithm 1. This algorithm is used to transform one signal that passes through a set of points P, called control points. The transformation is defined by a new set of control points Q, which are the transformed positions of the control points P [7]. The control points Q are obtained, in our implementation, by applying transformation fields F translations to the original control points P. MLS applies the transformation by initially creating one local approximation function l_v for each point v of the time series. Thus, for every point we solve the best affine transformations that approximates the new control points Q [line 3 of Alg 1]. There is a weight decay in the importance of the control points compared to the point for which we are defining a local transformation, approximation of

Algorithm 1. MLS [7]

```
Require: A series: $S$, A transformation field: $F$, Control Points: $P$
Ensure: New transformed instance: $T$
1: $Q \leftarrow [P_1 + F_1, P_2 + F_2, ...]$
2: for $v = 1$ to $|S|$ do
3:     Search $l_v$ that minimizes
       $\mathrm{argmin}_{l_v} \sum_{i=1}^{|P|} w_i |l_v(P_i) - Q_i|^2$ , where $w_i = \frac{1}{|P_i - v|^{2\alpha}}$
4:     $T[v] \leftarrow l_v(S[v])$
5: end for
6: return $T$
```

near control points get more impact. The speed of decay is controlled by a hyper parameter α.

Once the local transformation function l_v is computed, then the value at point v in the transformed series is computed by applying the searched transformation function over the value in the original series. In order for the transformed series to look as realistic compared to the original series, the transformation should be as rigid as possible, that is, the space of deformations should not even include uniform scaling, therefore we follow the rigid transformations optimization [7] in solving line 3 of Alg. 1. This subsection only introduced the transformation fields and the underlying algorithm used to transform a series, while the successive subsections will show how to search for the best magnitudes of the transformation fields vector elements, in order for the transformed instances to encapsulate intra-class variations.

3.4 Warping Maps

Before introducing the main hub of our method concerning how the variance-generating transformation fields are created, we initially need to present some necessary concepts and means, which are used in the successive subsection to analyze the intra-class variance.

DTW is an algorithm used to compute the similarity/distance between two time series. A cost matrix, denoted $\mathbf{W}$, is build progressively by computing the subtotal warping cost of aligning two series A and B. The cost is computed incrementally backwards until reaching a stopping condition in aligning the first points of the series. The overall cost is accumulatively computed at the topmost index value of the matrix, whose indices correspond to the length of series.

$$\begin{aligned}\mathrm{DTW}(A, B) &= \mathbf{W}_{\mathrm{length}(\mathbf{A}),\mathrm{length}(\mathbf{B})}\\ \mathbf{W}_{1,1} &= (A_1 - B_1)^2\\ \mathbf{W}_{i,j} &= (A_i - B_j)^2 + \min(\mathbf{W}_{i-1,j}, \mathbf{W}_{i,j-1}, \mathbf{W}_{i-1,j-1}) \qquad (1)\end{aligned}$$

An optimal **warping path** between series A and B, defined as $\tau_{A,B}$ or here shortly τ, is a list of aligned indexes of points along the cost matrix. The sum of distances of the aligned points along the warping path should sum up to the exact distance cost of DTW. The list of the warping path indexes pairs (i, j) corresponds to the chain of the recursive calls in the cost computation $W_{i,j}$. The sum of the distances among the values of the aligned indexes of two series, yields the minimum distance, which is equal to the DTW formulation.

$$\tau(A, B) = \{(i, j) \mid W_{i,j} \text{ called in the chain of recursion of } DTW(A, B)\} \quad (2)$$

A **warping map**, denoted M, is a square matrix whose elements are built by overlapping the warping paths of all-vs-all instances in an equi-length time series dataset. In this overlapping context, a cell of the warping map matrix denotes how often a warping alignment occur at that indexe. Equation 3 formalizes the procedure of building a warping map as a superposition (frequency) of warping paths of all time series pairs A, B from dataset S.

$$\mathbf{M}(i, j) \leftarrow \mid \{(A, B) \in S^2 \mid (i, j) \in \tau(A, B)\} \mid \quad (3)$$

A filtered warping map is created similarly as shown in Algorithm 2, where we filter only those warping paths that are either right or left aligned at a specific point. For instance, if we need to filter for right alignment at a point P, we need to build the DTW warping of any two series pairs, denoted τ, and then check if the aligned index at the second series is higher than (right of) the index on the first series. For instance, the notation $\tau(A, B)_P$ denotes the aligned index at series B corresponding to time P of first series A.

Algorithm 2. FilteredWarpingMap

```
Require: Dataset of time series $\mathbf{S}$, Control Point $P$, Direction $D$
Ensure: Filtered warping map $\mathbf{M}$
if $D = right$ then
    $\mathbf{M}(i, j) \leftarrow \mid \{(A, B) \in S^2 \mid (i, j) \in \tau(A, B) \wedge P < \tau(A, B)_P\} \mid$
else
    $\mathbf{M}(i, j) \leftarrow \mid \{(A, B) \in S^2 \mid (i, j) \in \tau(A, B) \wedge P \geq \tau(A, B)_P\} \mid$
end if
return $\mathbf{M}$
```

In our forthcoming analysis we build warping paths by providing a filtered dataset of instances belonging to only one class. Therefore we will construct one warping map per class.

3.5 Variance Distribution Analysis and Creation of Transformation Fields

In this section we present the main method of creating the transformation, which is be based on the analysis of the variance distributions of warping paths. The transformation fields represent local perturbation vectors of the predefined regions, R many, by translating the representative control points. Each control point/region is translated both left and right, therefore creating $2 \times R$ total transformation fields. The amount of translational transformation to be applied to every region is computed by making use of the warping maps. For each region, R_i, we filter in turn the right and left warping paths of instance warping alignments at that region, in order to analyze the general left/right translational variances of the warping alignments on other regions as an impact of a transformation at R_i. An illustration is found on Figure 3. The time series is divided into three regions defined by their centroid control points. In the image, part b), c), d) we show the filtered warping maps for the right warping alignments at every control points. Please note that three more filtered warping maps could be created for left alignments, but are avoided due to lack of space.

Once we built the warping map, we can successively construct the distribution of the warped alignments at every control points. For every region where we apply left/right translational perturbations, the *transformation field is created to be equal to the means of the warped points*, as an impact of the perturbation. The means are selected as transformation field, because they represents the tendency of variations at every control point. An illustration of the distributions is depicted in Figure 4. Only two distribution plots are shown belonging to the warping maps in Figure 3, part **b)** and **c)**.

Algorithm 3. ComputeTransformationFields

```
Require: Dataset of time series: $S$, A class: $label$, A list of control points CP
Ensure: List of transformation fields: $L$
 1: $L \leftarrow \emptyset$
 2: $S_{label} \leftarrow \{A \in S \mid A$ has class $label\}$
 3: for $P \in$ CP do
 4:     for direction $\in \{\text{left}, \text{right}\}$ do
 5:         $M \leftarrow$FilteredWarpingMap($S_{label}, P$, direction) from Algorithm 2
 6:         for $P' \in$ CP do
 7:             $F_j \leftarrow \frac{1}{\|M_{j,*}\|} \sum_{k=1}^{|S_{label}|} (M_{j,k} \cdot (k - P'))\ ,\ \ j \in [1...|CP|]$
 8:         end for
 9:         $L \leftarrow L \cup \{F\}$
10:     end for
11: end for
12: return $L$
```

For instance, the means of the distributions, which also form transformation fields at image a), represents the warping distributions as an impact of perturbation of R_1 are [34 24 0]. We can conclude that a right perturbation at R_1 causes a right translational impact on R_2, but fades away at R_3. Therefore the transformations of instances at R_1, will be in proportional to this distribution. Similarly in image b) there is a perturbation on R_2, which has a stronger impact on R_1 than on R_3.

The Algorithm 3 describes the creation of transformation fields. For every control point [line 3], we analyze the right and left variations [line 4] and get respective filtered warping maps [line 5]. The impact of such variation on other control points [line 6] is taken into consideration by the weighted mean variance at each other control point [line 7].

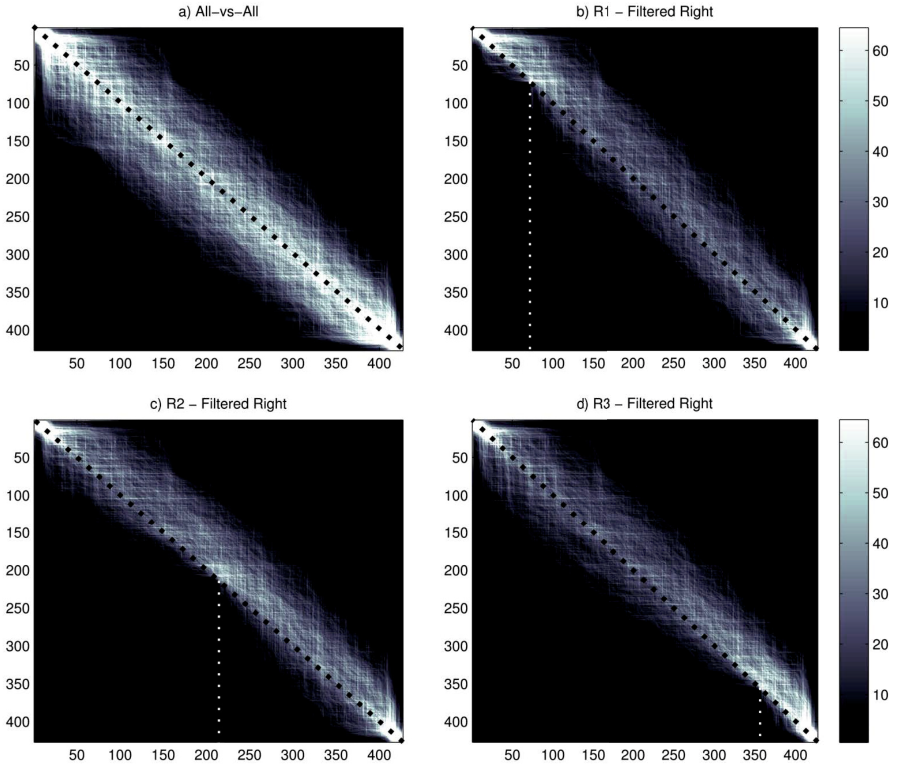


Fig. 3. An illustration concerning the warping map belonging to label 0 of the *OSULeaf* dataset. The time series are divided into three region R_1, R_2, R_3 for analysis. The center of each region is defined as a control point on time indices [71,213,355]. **a)** All-vs-all warping map. **b)** Warping map created by filtering only right warping paths at control point of R_1 at index 71. **c)** Warping map at control point of R_2 at index 213. **d)** A similar right warping filter of R_3 at 355.

3.6 Learning and Virtual Support Vectors

The defined transformation fields are used during the classification of time series. Even though in principle various classifiers can benefit from larger training set, still transforming all instances deteriorates the learning time of methods. SVMs have a crucial advantage because they point out the important instances (support vectors) which are needed to be transformed. In our study only the support vectors of a SVM model are transformed, called Virtual Support Vectors (VSV) [5]. Such selective approach ensures that the decision boundary is redefined only by instances close to it, hence the support vectors. The training set is extended to include MLS transformations of the support vectors as shown in Equation 4.

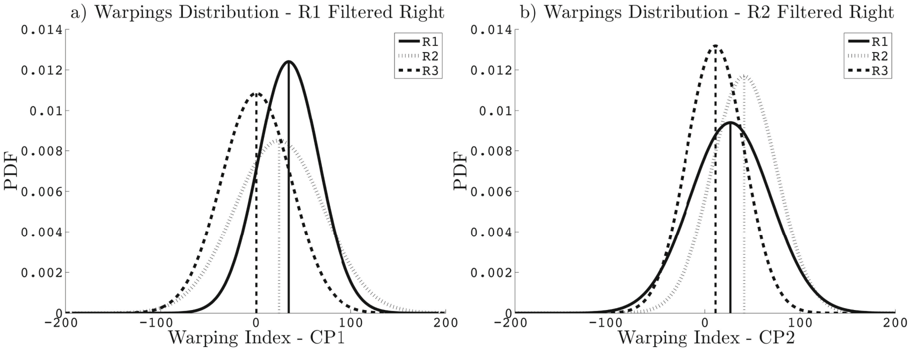


Fig. 4. Approximation with Gaussian Probability Density Functions (PDF) regarding the warping distribution of filtered warping maps in Figure 3, images b and c. **a)** The warping distribution as a result of right-warping occuring at R1/CP1. **b)** The warping distribution as a result of right-warping occuring at R2/CP2.

Given a list of control points, denoted CP, and a list of transformation fields, denoted TF, a transformation scale factor μ, then Equation 4 represents the addition of transformed support vectors obtained by building a model, denoted $svmModel$, from the training set S_{train}.

$$\begin{aligned} S^*_{train} = S_{train} \cup \{ \; & MLS\ (sv, \mu \cdot v, \text{CP}) \mid sv \in supportVectors(svmModel) \\ & \wedge \quad v \in TF_{label(sv)} \} \end{aligned} \tag{4}$$

Algorithm 4 describes the classification procedure in pseudo code style. The values of the transformation scales are computed by hyper-parameter search on a validation split search during the experiments.

4 Experimental Setup

In order to evaluate the performance of our proposed method, denoted as Invariant SVM, or shortly ISVM, we implemented and evaluated the following set of baselines:

- **SVM:** The default SVM is a natural choice for a baseline. The performance of our method compared to the standard SVM will give us indications on the success of redefining the decision boundaries, by injecting transformed support vectors.
- **DTW-NN:** Characterized as a hard-to-beat baseline in time series classification, which has been reported to achieve hard-to-beat classification accuracy [15]. The relative performance of our method compared to DTW-NN will give hints whether a refined maximum-margin is competitive, or not.

Algorithm 4. LearnModel

```
Require: Training set of time series: $S_{train}$, SVM hyper parameters: $\theta$, Transformation
    Fields: $TF$, Control Points: $CP$, Transformation Scale: $\mu$
Ensure: SVM model: $svmModel$
$svmModel \leftarrow svm.train(S_{train}, \theta)$
for $sv \in supportVectors(svmModel)$ do
    $label \leftarrow sv.label$
    $TF_{label} \leftarrow$ Field vectors of $TF$ for class $label$
    for $v \in TF_{label}$ do
        $VSV \leftarrow MLS(sv, \mu \times v, CP)$ from Algorithm 1
        $S_{train} \leftarrow S_{train} \cup \{VSV\}$
    end for
end for
$svmModel \leftarrow svm.train(S_{train}, \theta)$
return $svmModel$
```

The UCR collection of time series dataset was selected for experimentation. Very few large datasets whose transformation fields creation was excessively time consuming were omitted. All the datasets were randomly divided into five subsets/folds of same size (5-folds cross-validation). Each random subset was stratified, meaning that, the number of instances per label was kept equal on all subsets. In turn each fold was used once for testing the method, while three out of the remaining four for training and one for validation. The inhomogeneous polynomial kernel, $k(x, y) = (\gamma\, x \cdot y + 1)^d$, was applied for both the standard SVM as well as our method. The degree d was found to perform overall optimal at value of 3. A hyper parameter search was conducted in order to select the optimal values of the kernel's parameter γ and the methods transformation scale μ of Algorithm 4, by searching for maximum performance on the validation set after building the model on the train set. SVM's parameter C was searched among $\{0.25, 0.5, 1, 2, 4\}$. The number of regions/control points was found to be optimal around 10. The performance is finally tested with a cross-validation run over all the splits[2].

5 Results

A cumulative results table involving the experimentation results is found in Table 1. For every dataset, the mean and standard deviations of the cross validation error rates is reported in columns. The non-overlapping 1-σ confidence interval results, representing significance of ISVM/SVM results versus DTW-NN, are annotated with a circle (°). We grouped the datasets into two categories, easy ones and challenging ones. The criteria of the split is based on an error rate threshold, where values greater than 5% of the default SVM are grouped as easy dataset. The values in bold indicate the best mean error rate for the respective

[2] The authors provide the source code upon request.

dataset/row. The last row indicates a sum of the wins for each method, where draw points are split to ties. In brackets we denote the wins with significant and non-significant intervals.

The first message of the experiments is that the performance of our method is improving the accuracy of a standard SVM. In various cases like 50words, Cricket_X, Cricket_Y, Cricket_Z, Lighting7, OSULeaf, WordsSynonyms, the improvement is very significant ranging from +5% up to +11% accuracy. Thus it is appropriate to use our method for boosting the SVM accuracy, without adding noise.

The second and more important message is that our method produces better mean error rates than DTW-NN, winning on the majority of the datasets. The performance on the easy datasets is even (7 to 7). However, our method outperforms DTW-NN on the majority (11 to 5) of the challenging datasets. The invariant SVM looses significantly only on Cricket_* and Lighting2. In contrast, it significantly outperforms DTW-NN on 50words, Fish, OliveOil, SwedishLeaf, uWaveGestureLibrary_* and WordsSynonyms. Thus, our experiments demonstrate the superiority of our approach to DTW-NN.

The transformation scale parameter introduced in Algorithm 3, controls the scale of the transformation fields perturbations to be applied to the instances. Intuitively, optimal transformation fields redefine the decision boundary, while excessive magnitudes of transformations deteriorate into noisy instances. A demonstration of the transformation fields' scale parameter behavior is presented in Figure 5.

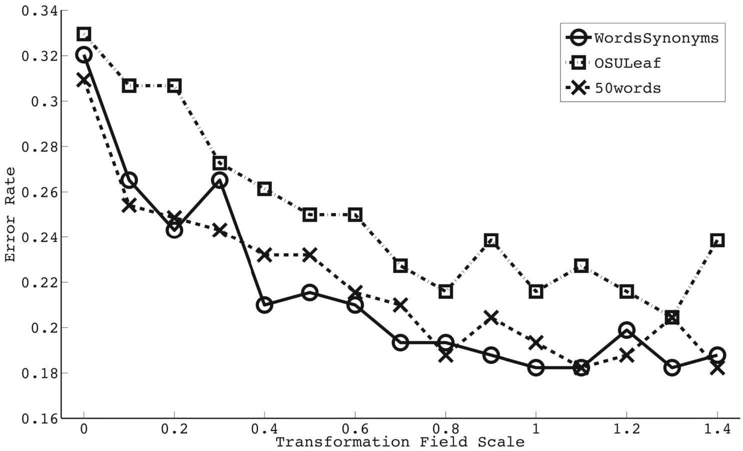


Fig. 5. The effect of increasing the transformation field scale on three typical datasets. The accuracy improves proportionally with the scale, until a minimum point is reached. After the optimal error rate, the large transformations produce noise and deteriorate accuracy, as for instance in OSULeaf, after minimum at scale value 1.3.

Table 1. 5-fold Cross-Validation Experiments Results - Error Rate Fractions

(i) **ISVM** *: Our proposed method Invariant Support Vector Machines*
(ii) **DTW-NN** *: Nearest Neighbor with Dynamic Time Warping*
(iii) **SVM***: The default Support Vector Machines*

Dataset	**ISVM**		**DTW-NN**		**SVM**	
	mean	*st.dev.*	*mean*	*st.dev.*	*mean*	*st.dev.*
Easy Datasets						
CBF	0.002	0.003	**0.000**	0.000	0.002	0.003
Coffee	°**0.000**	0.000	0.073	0.010	°**0.000**	0.000
DiatomSizeReduction	°**0.000**	0.000	0.006	0.000	°**0.000**	0.000
ECGFiveDays	°**0.001**	0.003	0.007	0.000	°**0.001**	0.003
FaceAll	**0.020**	0.007	0.024	0.000	0.027	0.005
FaceFour	**0.036**	0.038	0.054	0.006	0.045	0.056
FacesUCR	°**0.021**	0.007	0.031	0.000	0.026	0.010
Gun_Point	°**0.030**	0.027	0.075	0.001	0.050	0.040
ItalyPowerDemand	°**0.026**	0.019	0.051	0.000	°**0.026**	0.019
MoteStrain	0.050	0.016	**0.045**	0.000	0.060	0.020
SonyAIBORobotSurface	°**0.008**	0.011	0.027	0.000	°**0.008**	0.011
SonyAIBORobotSurfaceII	0.004	0.004	0.029	0.000	°**0.003**	0.005
Symbols	0.027	0.016	**0.019**	0.000	0.029	0.012
synthetic_control	0.020	0.013	°**0.007**	0.000	0.022	0.015
Trace	0.030	0.027	°**0.000**	0.000	0.045	0.048
TwoLeadECG	0.002	0.002	**0.001**	0.000	0.002	0.002
Two_Patterns	0.001	0.001	°**0.000**	0.000	0.006	0.001
wafer	°**0.002**	0.002	0.006	0.000	°**0.002**	0.001
Wins (Sig./Non-Sig.)	**7 (5/2)**		**7 (3/4)**		**4 (4/0)**	
Challenging Datasets						
50words	°**0.199**	0.008	0.287	0.001	0.272	0.035
Adiac	°**0.202**	0.038	0.341	0.001	0.206	0.037
Beef	°**0.267**	0.109	0.467	0.009	°**0.267**	0.109
Cricket_X	0.300	0.025	°**0.197**	0.001	0.391	0.033
Cricket_Y	0.271	0.021	°**0.213**	0.001	0.388	0.045
Cricket_Z	0.306	0.043	°**0.188**	0.000	0.399	0.031
ECG200	**0.110**	0.052	0.160	0.003	0.125	0.040
Fish	°**0.094**	0.031	0.211	0.000	0.103	0.031
Lighting2	0.289	0.025	°**0.099**	0.002	0.297	0.013
Lighting7	**0.252**	0.054	0.285	0.010	0.357	0.068
OliveOil	**0.083**	0.083	0.133	0.006	**0.083**	0.083
OSULeaf	0.296	0.039	**0.285**	0.003	0.346	0.059
SwedishLeaf	°**0.079**	0.017	0.185	0.001	0.082	0.014
uWaveGestureLibrary_X	°**0.193**	0.012	0.251	0.000	0.197	0.013
uWaveGestureLibrary_Y	°**0.253**	0.009	0.342	0.000	0.259	0.009
uWaveGestureLibrary_Z	°**0.249**	0.008	0.301	0.000	0.256	0.011
WordsSynonyms	°**0.200**	0.038	0.270	0.000	0.261	0.027
Wins (Sig./Non-Sig.)	**11 (9/2)**		**5 (4/1)**		**1 (0.5/0.5)**	

Finally, it is worth mentioning that the time complexity of our method is obviously worse than that of a normal SVM, because of the enlarged training set. Yet, the computational time is not prohibitive in terms of run-time feasibility. In Table 2 you can find some test run times in minutes, of typical prototypes of easy and challenging datasets, with the aim of demonstrating the run time feasibility of our method compared to DTW-NN. The run-time minutes shown on the last column are measured over the same random dataset fold.

Table 2. Classification Run Times of Typical Dataset

Dataset	**# labels**	**# instances**	**length**	**ISVM** Time(min)
Coffee	2	56	286	0.01
ECG200	2	200	96	0.04
wafer	2	7174	152	5.48
FacesUCR	14	2250	131	8.73
50words	50	905	270	14.17
Cricket_X	12	780	300	24.00

6 Conclusion

This study we introduced a novel instance transformation method, which is used to boost the performance of SVM via transforming the support vectors. The proposed method utilizes the distribution of warping variances, yield from warping alignment maps, in order to define transformation fields, which represent variances at a predefined set of local regions of a particular class. Therefore the virtual support vectors which are generated by applying the transformation fields, represent the necessary intraclass variation and redefines the maximum margin decision boundary. The superiority of our method is demonstrated by extensive experimentations on 35 datasets of the UCR collection. In a group of easy datasets, the presented method is on a par competitive to the baselines, while being clearly superior on a set of challenging datasets.

As a planned future work, we will focus on analyzing the joint operation of our method with possible inclusion of efficient similarity based metric into the SVM kernel. In parallel, other state-of-art invariant classifiers will be explored, in particular Convolutional Neural Networks.

References

1. Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications With R Examples. Springer (2011) ISBN 978-1-4419-7864-6
2. Keogh, E.J., Pazzani, M.J.: Scaling up dynamic time warping for datamining applications. In: KDD, pp. 285–289 (2000)
3. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.J.: Querying and mining of time series data: experimental comparison of representations and distance measures. PVLDB 1(2), 1542–1552 (2008)

4. Gudmundsson, S., Runarsson, T.P., Sigurdsson, S.: Support vector machines and dynamic time warping for time series. In: IJCNN, pp. 2772–2776. IEEE (2008)
5. Schölkopf, B., Burges, C., Vapnik, V.: Incorporating invariances in support vector learning machines, pp. 47–52. Springer (1996)
6. DeCoste, D., Schölkopf, B.: Training invariant support vector machines. Machine Learning 46(1-3), 161–190 (2002)
7. Schaefer, S., McPhail, T., Warren, J.D.: Image deformation using moving least squares. ACM Trans. Graph. 25(3), 533–540 (2006)
8. Kehagias, A., Petridis, V.: Predictive modular neural networks for time series classification. Neural Networks 10(1), 31–49 (1997)
9. Nanopoulos, A., Alcock, R., Manolopoulos, Y.: Feature-based classification of time-series data. International Journal of Computer Research 10, 49–61 (2001)
10. Pavlovic, V., Frey, B.J., Huang, T.S.: Time-series classification using mixed-state dynamic bayesian networks. In: CVPR, p. 2609. IEEE Computer Society (1999)
11. Rodríguez, J.J., Alonso, C.J.: Interval and dynamic time warping-based decision trees. In: Proceedings of the 2004 ACM Symposium on Applied Computing, SAC 2004, pp. 548–552. ACM, New York (2004)
12. Rodríguez, J.J., Alonso, C.J., Boström, H.: Learning First Order Logic Time Series Classifiers: Rules and Boosting. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 299–308. Springer, Heidelberg (2000)
13. Kim, S., Smyth, P., Luther, S.: Modeling waveform shapes with random effects segmental hidden markov models. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, UAI 2004, Arlington, Virginia, United States, pp. 309–316. AUAI Press (2004)
14. Mizuhara, Y., Hayashi, A., Suematsu, N.: Embedding of time series data by using dynamic time warping distances. Syst. Comput. Japan 37(3), 1–9 (2006)
15. Xi, X., Keogh, E.J., Shelton, C.R., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: ICML. ACM International Conference Proceeding Series, vol. 148, pp. 1033–1040. ACM (2006)
16. Keogh, E.J., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. Knowl. Inf. Syst. 7(3), 358–386 (2005)
17. Niyogi, P., Girosi, F., Poggio, T.: Incorporating prior information in machine learning by creating virtual examples. Proceedings of the IEEE, 2196–2209 (1998)
18. Loosli, G., Canu, S., Vishwanathan, S.V.N., Smola, A.J.: Invariances in classification: an efficient svm implementation. In: Proceedings of the 11th International Symposium on Applied Stochastic Models and Data Analysis (2005)
19. Loosli, G., Canu, S., Bottou, L.: Training invariant support vector machines using selective sampling. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large Scale Kernel Machines, pp. 301–320. MIT Press, Cambridge (2007)
20. Shimodaira, H.: Noma, K.-i., Nakai, M., Sagayama, S.: Support vector machine with dynamic time-alignment kernel for speech recognition. In: Dalsgaard, P., Lindberg, B., Benner, H., Tan, Z.H. (eds.) INTERSPEECH, ISCA, pp. 1841–1844 (2001)
21. Bahlmann, C., Haasdonk, B., Burkhardt, H.: On-line handwriting recognition with support vector machines - a kernel approach. In: Proc. of the 8th IWFHR, pp. 49–54 (2002)
22. Zhang, D., Zuo, W., Zhang, D., Zhang, H.: Time series classification using support vector machine with gaussian elastic metric kernel. In: ICPR, pp. 29–32. IEEE (2010)
23. Rahimi, A., Recht, B., Darrell, T.: Learning to transform time series with a few examples. IEEE Trans. Pattern Anal. Mach. Intell. 29(10), 1759–1775 (2007)

Learning Bi-clustered Vector Autoregressive Models

Tzu-Kuo Huang and Jeff Schneider

School of Computer Science, Carnegie Mellon University
{tzukuoh,schneide}@cs.cmu.edu

Abstract. Vector Auto-regressive (VAR) models are useful for analyzing temporal dependencies among multivariate time series, known as *Granger causality*. There exist methods for learning sparse VAR models, leading directly to causal networks among the variables of interest. Another useful type of analysis comes from clustering methods, which summarize multiple time series by putting them into groups. We develop a methodology that integrates both types of analyses, motivated by the intuition that Granger causal relations in real-world time series may exhibit some clustering structure, in which case the estimation of both should be carried out together. Our methodology combines sparse learning and a nonparametric *bi-clustered* prior over the VAR model, conducting full Bayesian inference via blocked Gibbs sampling. Experiments on simulated and real data demonstrate improvements in both model estimation and clustering quality over standard alternatives, and in particular biologically more meaningful clusters in a T-cell activation gene expression time series dataset than those by other methods.

Keywords: time-series analysis, vector auto-regressive models, bi-clustering, Bayesian non-parametrics, gene expression analysis.

1 Introduction

Vector Auto-regressive (VAR) models are standard tools for analyzing multivariate time series data, especially their temporal dependencies, known as *Granger causality*[1] [7]. VAR models have been successfully applied in a number of domains, such as finance and economics [23,14], to capture and forecast dynamic properties of time series data. Recently, researchers in computational biology, using ideas from sparse linear regression, developed sparse estimation techniques for VAR models [5,11,22] to learn from high-dimensional genomic time series a small set of pairwise, directed interactions, referred to as gene regulatory networks, some of which lead to novel biological hypotheses.

While individual edges convey important information about interactions, it is often desirable to obtain an aggregate and more interpretable description of the network of interest. One useful set of tools for this purpose are graph clustering methods [20], which identify groups of nodes or vertices that have similar types

[1] More precisely, *graphical* Granger causality for more than two time series.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 741–756, 2012.

of connections, such as a common set of neighboring nodes in undirected graphs, and shared parent or child nodes in directed graphs. These methods have been applied in the analysis of various types of networks, such as [6], and play a key role in graph visualization tools [9].

Motivated by the wide applicability of the above two threads of work and the observation that their goals are tightly coupled, we develop a methodology that integrates both types of analyses, estimating the underlying Granger causal network and its clustering structure *simultaneously.* We consider the following first-order p-dimensional VAR model:

$$\mathbf{x}_{(t)} \;=\; \mathbf{x}_{(t-1)}A + \boldsymbol{\epsilon}_{(t)}, \quad \boldsymbol{\epsilon}_{(t)} \;\sim\; \mathcal{N}(\mathbf{0}, \sigma^2 I), \tag{1}$$

where $\mathbf{x}_{(t)} \in \mathbb{R}^{1\times p}$ denotes the vector of variables observed at time t, $A \in \mathbb{R}^{p\times p}$ is known as the transition matrix, whose non-zero entries encode Granger causal relations among the variables, and $\boldsymbol{\epsilon}_{(t)}$'s denote independent noise vectors drawn from a zero-mean Gaussian with a spherical covariance $\sigma^2 I$. Our goal is to obtain a transition matrix estimate $\widehat{A}$ that is both *sparse*, leading directly to a causal network, and *clustered* so that variables sharing a similar set of connections are grouped together. Since the rows and the columns of A indicate different roles of the variables, the former revealing how variables affect themselves and the latter showing how variables get affected, we consider the more general *bi-clustering* setting, which allows two different sets of clusters for rows and columns, respectively. We take a nonparametric Bayesian approach, placing over A a nonparametric bi-clustered prior and carrying out full posterior inferences via a blocked Gibbs sampling scheme. Our simulation study demonstrates that when the underlying VAR model exhibits a clear bi-clustering structure, our proposed method improves over some natural alternatives, such as adaptive sparse learning methods [24] followed by bi-clustering, in terms of model estimation accuracy, clustering quality, and forecasting capability. More encouragingly, on a real-world T-cell activation gene expression time series data set [18] our proposed method finds an interesting bi-clustering structure, which leads to a biologically more meaningful interpretation than those by some state-of-the art time series clustering methods.

Before introducing our method, we briefly discuss related work in Section 2. Then we define our bi-clustered prior in Section 3, followed by our sampling scheme for posterior inferences in Section 4. Lastly, we report our experimental results in Section 5 and conclude with Section 6.

2 Related Work

There has been a lot of work on sparse estimation of causal networks under VAR models, and perhaps even more on graph clustering. However, to the best of our knowledge, none of them has considered the simultaneous learning scheme we propose here. Some of the more recent sparse VAR estimation work [11,22] takes into account dependency further back in time and can even select the right length of history, known as the order of the VAR model. While developing our method around first-order VAR models, we observe that it can also learn higher-order bi-clustered models by, for example, assigning transition matrix entries across multiple time lags to the same bi-cluster.

Another large body of related work ([13,16,2], just to name a few) concerns bi-clustering (or co-clustering) a data matrix, which usually consists of relations between two sets of objects, such as user ratings on items, or word occurrences in documents. Most of this work models data matrix entries by mixtures of distributions with different *means*, representing, for example, different mean ratings by different user groups on item groups. In contrast, common regularization schemes or prior beliefs for VAR estimation usually assume zero-mean entries for the transition matrix, biasing the final estimate towards being stable. Following such a practice, our method models transition matrix entries as *scale mixtures* of zero-mean distributions.

Finally, clustering time series data has been an active research topic in a number of areas, in particular computational biology. However, unlike our Granger causality based bi-clustering method, most of the existing work, such as [17,3] and the references therein, focus on grouping together *similar* time series, with a wide range of similarity measures from simple linear correlation to complicated Gaussian process based likelihood scores. Differences between our method and existing similarity-based approaches are demonstrated in Section 5 through both simulations and experiments on real data.

3 Bi-clustered Prior

We treat the transition matrix $A \in \mathcal{R}^{p \times p}$ as a random variable and place over it a "bi-clustered" prior, as defined by the following generative process:

$$\begin{aligned} \boldsymbol{\pi}_u &\sim \mathsf{Stick\text{-}Break}(\alpha_u), \qquad & \boldsymbol{\pi}_v &\sim \mathsf{Stick\text{-}Break}(\alpha_v), \\ \{u_i\}_{1\le i\le p} &\overset{i.i.d}{\sim} \mathsf{Multinomial}(\boldsymbol{\pi}_u), \qquad & \{v_j\}_{1\le j\le p} &\overset{i.i.d}{\sim} \mathsf{Multinomial}(\boldsymbol{\pi}_v), \\ & & \{\lambda_{kl}\}_{1\le k,l\le\infty} &\overset{i.i.d.}{\sim} \mathsf{Gamma}(h,c), \qquad (2) \\ & & A_{ij} &\sim \mathsf{Laplace}(0, 1/\lambda_{u_i v_j}), \quad 1 \le i,j \le p. \qquad (3) \end{aligned}$$

The process starts by drawing row and column mixture proportions $\boldsymbol{\pi}_u$ and $\boldsymbol{\pi}_v$ from the "stick-breaking" distribution [21], denoted by $\mathsf{Stick\text{-}Break}(\alpha)$ and defined on an infinite-dimensional simplex as follows:

$$\begin{aligned} \beta_k &\sim \mathsf{Beta}(1,\alpha), \\ \pi_k &:= \beta_k \prod_{m<k} (1-\beta_m), \quad 1 \le k \le \infty, \end{aligned} \qquad (4)$$

where $\alpha > 0$ controls the average length of pieces broken from the stick, and may take different values α_u and α_v for rows and columns, respectively. Such a prior allows for an infinite number of mixture components or clusters, and lets the data decide the number of *effective* components having positive probability masses, thereby increasing modeling flexibility. The process then samples row-cluster and

Algorithm 1. Blocked Gibbs Sampler

```
Input: Data $X$ and $Y$, hyper-parameters $h, c, \alpha_u, \alpha_v$, and initial values $A^{(0)}, L^{(0)}$, $\mathbf{u}^{(0)}$, $\mathbf{v}^{(0)}$, $(\sigma^{(0)})^2$
Output: Samples from the full joint posterior $p(A, L, \mathbf{u}, \mathbf{v}, \sigma^2 \mid X, Y)$
Set iteration $t = 1$
repeat
  for $i = 1$ to $p$ do
    $A_i^{(t)} \sim p(A_i \mid A_{1:(i-1)}^{(t)}, A_{(i+1):p}^{(t-1)}, \mathbf{u}^{(t-1)}, \mathbf{v}^{(t-1)}, (\sigma^{(t-1)})^2), L^{(t-1)}, X, Y)$
  end for
  for $i = 1$ to $p$ do
    $u_i^{(t)} \sim p(u_i \mid A^{(t)}, \mathbf{u}_{1:(i-1)}^{(t)}, \mathbf{u}_{(i+1):p}^{(t-1)}, \mathbf{v}^{(t-1)}, (\sigma^{(t-1)})^2, L^{(t-1)}, X, Y)$
  end for
  for $j = 1$ to $p$ do
    $v_j^{(t)} \sim p(v_j \mid A^{(t)}, \mathbf{u}^{(t)}, \mathbf{v}_{1:(j-1)}^{(t)}, \mathbf{v}_{(j+1):p}^{(t-1)}, (\sigma^{(t-1)})^2, L^{(t-1)}, X, Y)$
  end for
  $(\sigma^{(t)})^2 \sim p(\sigma^2 \mid A^{(t)}, \mathbf{u}^{(t)}, \mathbf{v}^{(t)}, L^{(t-1)}, X, Y)$
  $L^{(t)} \sim p(L \mid A^{(t)}, \mathbf{u}^{(t)}, \mathbf{v}^{(t)}, (\sigma^{(t)})^2, X, Y)$
  Increase iteration $t$
until convergence
Notations: superscript $(t)$ denotes iteration, $A_i$ denotes the $i$-th row of $A$, $A_{i:j}$ denotes the sub-matrix in $A$ from the $i$-th until the $j$-th row, and $\mathbf{u}_{i:j}$ denotes $\{u_n\}_{i \leq n \leq j}$.
```

column-cluster indicator variables u_i's and v_j's from mixture proportions $\boldsymbol{\pi}_u$ and $\boldsymbol{\pi}_v$, and for the k-th row-cluster and the l-th column-cluster draws an inverse-scale, or rate parameter λ_{kl} from a Gamma distribution with shape parameter h and scale parameter c. Finally, the generative process draws each matrix entry A_{ij} from a zero-mean Laplace distribution with inverse scale $\lambda_{u_i v_j}$, such that entries belonging to the same bi-cluster share the same inverse scale, and hence represent interactions of similar *magnitudes*, whether positive or negative.

The above bi-clustered prior subsumes a few interesting special cases. In some applications researchers may believe the clusters should be symmetric about rows and columns, which corresponds to enforcing $\mathbf{u} = \mathbf{v}$. If they further believe that within-cluster interactions should be stronger than between-cluster ones, they may adjust accordingly the hyper-parameters in the Gamma prior (2), or as in the group sparse prior proposed by [12] for Gaussian precision estimation, simply require all within-cluster matrix entries to have the same inverse scale constrained to be smaller than the one shared by all between-cluster entries. Our inference scheme detailed in the next section can be easily adapted to all these special cases.

There can be interesting generalizations as well. For example, depending on the application of interest, it may be desirable to distinguish positive interactions from negative ones, so that a bi-cluster of transition matrix entries possess not only similar strengths, but also *consistent signs.* However, such a generalization requires a more delicate per-entry prior and therefore a more complex sampling scheme, which we leave as an interesting direction for future work.

4 Posterior Inference

Let L denote the collection of λ_{kl}'s, $\mathbf{u}$ and $\mathbf{v}$ denote $\{u_i\}_{1\le i\le p}$ and $\{v_j\}_{1\le j\le p}$, respectively. Given one or more time series, collectively denoted as matrices X and Y whose rows represent successive pairs of observations, i.e.,

$$Y_i \;=\; X_i A + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \;\sim\; \mathcal{N}(\mathbf{0}, \sigma^2 I),$$

we aim to carry out posterior inferences about the transition matrix A, and row and column cluster indicators $\mathbf{u}$ and $\mathbf{v}$. To do so, we consider sampling from the full joint posterior $p(A, L, \mathbf{u}, \mathbf{v}, \sigma^2 \mid X, Y)$, and develop an efficient blocked Gibbs sampler outlined in Algorithm 1. Starting with some reasonable initial configuration, the algorithm iteratively samples rows of A, row and column-cluster indicator variables $\mathbf{u}$ and $\mathbf{v}$, the noise variance[2] σ^2, and the inverse scale parameters L from their respective conditional distributions. Next we describe in more details sampling from those conditional distributions.

4.1 Sampling the Transition Matrix A

Let A_{-i} denote the sub-matrix of A excluding the i-th row, X'_i and X'_{-i} denote the i-th column of X and the sub-matrix of X excluding the i-th column. Algorithm 1 requires sampling from the following conditional distribution:

$$p(A_i \mid A_{-i}, \mathbf{u}, \mathbf{v}, \sigma^2, L, X, Y) \;\propto\; \prod_{1\le j\le p} \mathcal{N}(A_{ij} \mid \mu_{ij}, \sigma_i^2)\mathsf{Laplace}(A_{ij} \mid 0, 1/\lambda_{u_i v_j}),$$

where

$$\mu_{ij} \;:=\; (X'_i/\|X'_i\|_2^2)^\top (Y - X'_{-i}A_{-i})'_j, \quad \sigma_i^2 \;:=\; \sigma^2/\|X'_i\|^2.$$

Therefore, all we need is sampling from univariate densities of the form:

$$f(x) \;\propto\; \mathcal{N}(x \mid \mu, \sigma^2)\mathsf{Laplace}(x \mid 0, 1/\lambda), \tag{5}$$

whose c.d.f. $F(x)$ can be expressed in terms of the standard normal c.d.f. $\Phi(\cdot)$:

$$F(x) \;=\; \frac{C_1}{C}\Phi\Big(\frac{x^- - (\mu+\sigma^2\lambda)}{\sigma}\Big) + \frac{C_2}{C}\Big(\Phi\Big(\frac{x^+ - (\mu-\sigma^2\lambda)}{\sigma}\Big) - \Phi\Big(-\frac{\mu-\sigma^2\lambda}{\sigma}\Big)\Big),$$

where $x^- := \min(x, 0)$, $x^+ := \max(x, 0)$, and

$$C := C_1\Phi\Big(-\frac{\mu+\sigma^2\lambda}{\sigma}\Big) + C_2\Big(1 - \Phi\Big(-\frac{\mu-\sigma^2\lambda}{\sigma}\Big)\Big),$$
$$C_1 := \frac{\lambda}{2}\exp\Big(\frac{\lambda(2\mu+\sigma^2\lambda)}{2}\Big), \quad C_2 := \frac{\lambda}{2}\exp\Big(\frac{\lambda(\sigma^2\lambda-2\mu)}{2}\Big).$$

We then sample from $f(x)$ with the inverse c.d.f. method. To reduce the potential sampling bias introduced by a fixed sampling schedule, we follow a random ordering of the rows of A in each iteration.

[2] Our sampling scheme can be easily modified to handle diagonal covariances.

Algorithm 1 generates samples from the full joint posterior, but sometimes it is desirable to obtain a point estimate of A. One simple estimate is the (empirical) posterior mean; however, it is rarely sparse. To get a sparse estimate, we carry out the following "sample EM" step after Algorithm 1 converges:

$$\widehat{A}^{\text{Biclus-EM}} := \arg\max_{A} \sum_{t} \log p(A \mid \mathbf{u}^{(t)}, \mathbf{v}^{(t)}, (\sigma^{(t)})^2, L^{(t)}, X, Y), \tag{6}$$

where t starts at a large number and skips some fixed number of iterations to give better-mixed and more independent samples. The optimization problem (6) is in the form of sparse least square regression, which we solve with a simple coordinate descent algorithm.

4.2 Sampling Row and Cluster Indicators

Since our sampling procedures for $\mathbf{u}$ and $\mathbf{v}$ are symmetric, we only describe the one for $\mathbf{u}$. It can be viewed as an instantiation of the general Gibbs sampling scheme in [13]. According to our model assumption, $\mathbf{u}$ is independent of the data X, Y and the noise variance σ^2 conditioned on all other random variables. Moreover, under the stick-breaking prior (4) over the row mixture proportions $\boldsymbol{\pi}_u$ and some fixed $\mathbf{v}$, we can view $\mathbf{u}$ and the rows of A as cluster indicators and samples drawn from a Dirichlet process mixture model with $\mathsf{Gamma}(h, c)$ as the base distribution over cluster parameters. Finally, the Laplace distribution and the Gamma distribution are conjugate pairs, allowing us to integrate out the inverse scale parameters L and derive the following "collapsed" sampling scheme:

$$\begin{aligned}
&p(u_i = k' \in \text{existing row-clusters} \mid A, \mathbf{u}_{-i}, \mathbf{v}) \\
&\propto \left(\prod_{k,l} \frac{\Gamma((N_{-i}[k] + \delta_{kk'})m_l + h)/(\Gamma(h)c^h)}{\left(\|A_{-i}[k,l]\|_1 + \delta_{kk'}\|A_i[l]\|_1 + 1/c \right)^{(N_{-i}[k]+\delta_{kk'})M[l]+h}} \right) \frac{N_{-i}[k']}{p - 1 + \alpha_u},
\end{aligned}$$

$$\begin{aligned}
&p(u_i = \text{a new row-cluster} \mid A, \mathbf{u}_{-i}, \mathbf{v}) \\
&\propto \left(\prod_{k,l} \frac{\Gamma(N_{-i}[k]M[l] + h)/(\Gamma(h)c^h)}{\left(\|A_{-i}[k,l]\|_1 + 1/c \right)^{N_{-i}[k]M[l]+h}} \cdot \frac{\Gamma(M[l] + h)/(\Gamma(h)c^h)}{\left(\|A_i[l]\|_1 + 1/c \right)^{M[l]+h}} \right) \frac{\alpha_u}{p - 1 + \alpha_u},
\end{aligned}$$

where $\Gamma(\cdot)$ is the Gamma function, δ_{ab} denotes the Kronecker delta function, $N_{-i}[k]$ is the size of the k-th row-cluster excluding A_i, $M[l]$ is the size of the l-th column-cluster, and

$$\|A_{-i}[k,l]\|_1 := \sum_{s \neq i, u_s = k, v_j = l} |A_{sj}|, \qquad \|A_i[l]\|_1 := \sum_{v_j = l} |A_{ij}|.$$

As in the previous section, we randomly permute u_i's and v_j's in each iteration to reduce sampling bias, and also randomly choose to sample $\mathbf{u}$ or $\mathbf{v}$ first.

Just as with the transition matrix A, we may want to obtain point estimates of the cluster indicators. The usual empirical mean estimator does not work here because the cluster labels may change over iterations. We thus employ the following procedure:

1. Construct a similarity matrix S such that

$$S_{ij} \;:=\; \frac{1}{T}\sum_{t} \delta_{u_i^{(t)} v_j^{(t)}}, \qquad 1 \leq i, j, \leq p,$$

   where t selects iterations to approach mixing and independence as in (6), and T is the total number of iterations selected.
2. Run normalized spectral clustering [15] on S, with the number of clusters set according to the spectral gap of S.

4.3 Sampling Noise Variance and Inverse Scale Parameters

On the noise variance σ^2 we place an inverse-Gamma prior with shape $a > 0$ and scale $\beta > 0$, leading to the following posterior:

$$\sigma^2 \mid A, X, Y \;\sim\; \mathsf{I\text{-}Gamma}(a + pT/2, 2\|Y - XA\|_F^{-2} + \beta), \tag{7}$$

where T is the number of rows in X and $\|\cdot\|_F$ denotes the matrix Frobenius norm. Due to the conjugacy mentioned in the last section, the inverse scale parameters λ_{kl}'s have the following posterior:

$$\lambda_{kl} \mid A, \mathbf{u}, \mathbf{v} \;\sim\; \mathsf{Gamma}(N[k]M[l] + h, (\|A[k,l]\|_1 + 1/c)^{-1}).$$

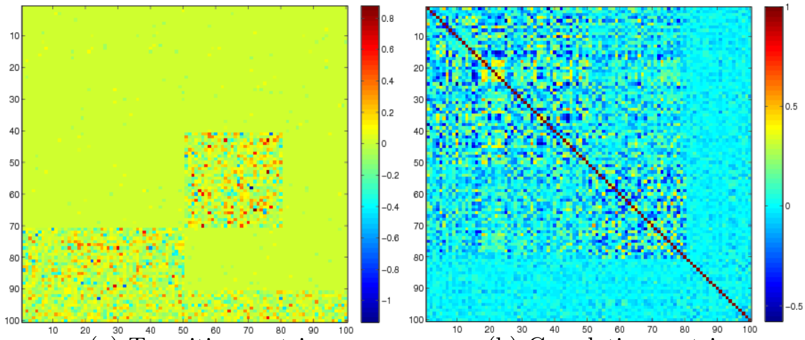
5 Experiments

We conduct both simulations and experiments on a real gene expression time series dataset, and compare the proposed method with two types of approaches:

Learning VAR by Sparse Linear Regression, Followed by Bi-clustering
Unlike the proposed method, which makes inferences about the transition matrix A and cluster indicators jointly, this natural baseline method first estimates the transition matrix by adaptive sparse or L_1 linear regression [24]:

$$\widehat{A}^{L_1} \;:=\; \arg\min_{A} \frac{1}{2}\|Y - XA\|_F^2 + \lambda \sum_{i,j} \frac{|A_{ij}|}{|\widehat{A}_{ij}^{\text{ols}}|^{\gamma}}, \tag{8}$$

where $\widehat{A}^{\text{ols}}$ denotes the ordinary least-square estimator, and then bi-clusters $\widehat{A}^{L_1}$ by either the cluster indicator sampling procedure in Section 4.2 or standard clustering methods applied to rows and columns separately. We compare the proposed method and this baseline in terms of predictive capability, clustering performance, and in the case of simulation study, model estimation error.



(a) Transition matrix (b) Correlation matrix

Fig. 1. Heat maps of the synthetic bi-clustered VAR

Clustering Based on Time Series Similarity
As described in Section 2, existing time series clustering methods are designed to group together time series that exhibit a similar behavior or dependency over time, whereas our proposed method clusters time series based on their (Granger) causal relations. We compare the proposed method with the time series clustering method proposed by [3], which models time series data by Gaussian processes and performs Bayesian Hierarchical Clustering [8], achieving state-of-the art clustering performances on the real genes time series data used in Section 5.

5.1 Simulation

We generate a transition matrix A of size 100 by first sampling entries in bi-clusters:

$$A_{ij} \sim \begin{cases} \mathsf{Laplace}(0, \sqrt{60}^{-1} i), & 41 \le i \le 70, 51 \le j \le 80, \\ \mathsf{Laplace}(0, \sqrt{70}^{-1}), & 71 \le i \le 90, 1 \le j \le 50, \\ \mathsf{Laplace}(0, \sqrt{110}^{-1}), & 91 \le i \le 100, 1 \le j \le 100, \end{cases} \tag{9}$$

and then all the remaining entries from a sparse back-ground matrix:

$$A_{ij} = \begin{cases} B_{ij} & \text{if } |B_{ij}| \ge q_{98}\big(\{|B_{i'j'}|\}_{1 \le i', j' \le 100}\big), \\ 0 & \text{otherwise,} \end{cases} \qquad i, j \text{ not covered in (9)},$$

where

$$\{B_{ij}\}_{1 \le i, j, \le 100} \overset{i.i.d.}{\sim} \mathsf{Laplace}(0, (5\sqrt{200})^{-1})$$

and $q_{98}(\cdot)$ denotes the 98-*th* percentile. Figure 1(a) shows the heat map of the actual A we obtain by the above sampling scheme, showing clearly four row-clusters and three column-clusters. This transition matrix has the largest eigenvalue modulus of 0.9280, constituting a stable VAR model.

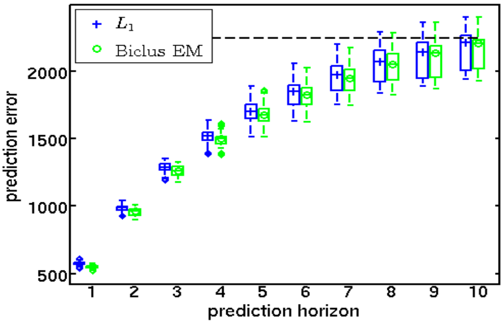


Fig. 2. Prediction errors up to 10 time steps. Errors for longer horizons are close to those by the mean (zero) prediction, shown in black dashed line, and are not reported.

We then sample 10 independent time series of 50 time steps from the VAR model (1), with noise variance $\sigma^2 = 5$. We initialize each time series with an independent sample drawn from the stationary distribution of (1), whose correlation matrix is shown in Figure 1(b), suggesting that clustering based on correlations among time series may not recover the bi-cluster structure in Figure 1(a).

To compare the proposed method with the two baselines described in the beginning of Section 5, we repeat the following experiment 20 times: a random subset of two time series are treated as testing data, while the other eight time series are used as training data. For L_1 linear regression (8) we randomly hold out two time series from the training data as a validation set for choosing the best regularization parameter λ from $\{2^{-2}, 2^{-1}, \ldots, 2^{10}\}$ and weight-adaption parameter γ from $\{0, 2^{-2}, 2^{-1}, \ldots, 2^2\}$, with which the final $\widehat{A}^{L_1}$ is estimated from all the training data. To bi-cluster $\widehat{A}^{L_1}$, we consider the following:

- **L$_1$+Biclus**: run the sampling procedure in Section 4.2 on $\widehat{A}^{L_1}$.
- **Refit+Biclus**: refit the non-zero entries of $\widehat{A}^{L_1}$ using least-square, and run the sampling procedure in Section 4.2.
- **L$_1$ row-clus** (**col-clus**): construct similarity matrices

$$S^u_{ij} := \sum_{1 \le s \le p} |\widehat{A}^{L_1}_{is}||\widehat{A}^{L_1}_{js}|, \quad S^v_{ij} := \sum_{1 \le s \le p} |\widehat{A}^{L_1}_{si}||\widehat{A}^{L_1}_{sj}|, \quad 1 \le i, j \le p.$$

Then run normalized spectral clustering [15] on S^u and S^v, with the number of clusters set to 4 for rows and 3 for columns, respectively.

For the second baseline, Bayesian Hierarchical Clustering and Gaussian processes (GPs), we use the R package BHC (version 1.8.0) with the squared-exponential covariance for Gaussian processes, as suggested by the author of the package. Following [3] we normalize each time series to have mean 0 and standard deviation 1. The package can be configured to use replicate information (multiple series) or not, and we experiment with both settings, abbreviated as BHC-SE

Table 1. Model estimation error on simulated data

	Normalized matrix error	Signed-support error
L_1	0.3133±0.0003	0.3012±0.0008
Biclus EM	**0.2419±0.0003**	**0.0662±0.0012**

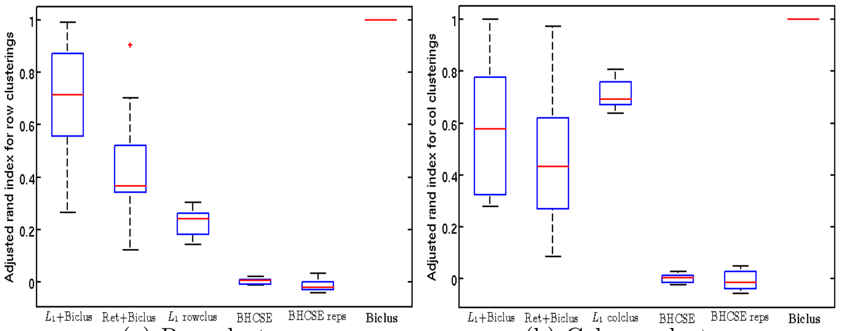
reps and BHC-SE, respectively. In both settings we give the BHC package the mean of the eight training series as input, but additionally supply BHC-SE reps a noise variance estimated from multiple training series to aid GP modeling.

In our proposed method, several hyper-parameters need to be specified. For the stick-breaking parameters α_u and α_v, we find that values in a reasonable range often lead to similar posterior inferences, and simply set both to be 1.5. We set the noise variance prior parameters in (7) to be $a = 9$ and $\beta = 10$. For the two parameters in the Gamma prior (2), we set $h = 2$ and $c = \sqrt{2p} = \sqrt{200}$ to bias the transition matrices sampled from the Laplace prior (3) towards being stable. Another set of inputs to Algorithm 1 are the initial values, which we set as follows: $A^{(0)} = \mathbf{0}$, $\mathbf{u}^{(0)} = \mathbf{v}^{(0)} = \mathbf{1}$, $(\sigma^{(0)})^2 = 1$, and $L^{(0)} = (h-1)c = \sqrt{200}$. We run Algorithm 1 and the sampling procedures for L_1+Biclus and Refit+Biclus for 2,500 iterations, and take samples in every 10 iterations starting from the 1,501-st iteration, at which the sampling algorithms have mixed quite well, to compute point estimates for A, $\mathbf{u}$ and $\mathbf{v}$ as described in Sections 4.1 and 4.2.

Figure 2 shows the squared prediction errors of L_1 linear regression (L_1) and the proposed method with a final sample EM step (Biclus EM) for various prediction horizons up to 10. Predictions errors for longer horizons are close to those by predicting the mean of the series, which is zero under our stable VAR model, and are not reported here. Biclus EM slightly outperforms L_1, and paired t tests show that the improvements for all 10 horizons are significant at a p-value ≤ 0.01. This suggests that when the underlying VAR model does have a bi-clustering structure, our proposed method can improve the prediction performance over adaptive L_1 regression, though by a small margin.

Another way to compare L_1 and Biclus EM is through model estimation error, and we report in Table 1 these two types of error:
Normalized matrix error: $\|\widehat{A} - A\|_F / \|A\|_F$,
Signed-support error: $\frac{1}{p^2} \sum_{1 \leq i,j \leq p} \mathbb{I}(\mathsf{sign}(\widehat{A}_{ij}) \neq \mathsf{sign}(A_{ij}))$.
Clearly, Biclus EM performs much better than L_1 in recovering the underlying model, and in particular achieves a huge gain in signed support error, thanks to its use of bi-clustered inverse scale parameters L.

Perhaps the most interesting is the clustering quality, which we evaluate by the *Adjusted Rand Index* [10], a common measure of similarity between two clusterings based on co-occurrences of object pairs across clusterings, with correction for chance effects. An adjusted Rand index takes the maximum value of 1 only when the two clusterings are identical (modulo label permutation), and is close to 0 when the agreement between the two clusterings could have resulted from two random clusterings. Figure 3 shows the clustering performances of different methods. The proposed method, labeled as Biclus, outperforms all alternatives greatly and always recovers the correct row and column clusterings. The



(a) Row clusters (b) Column clusters

Fig. 3. Adjusted Rand index on simulated data

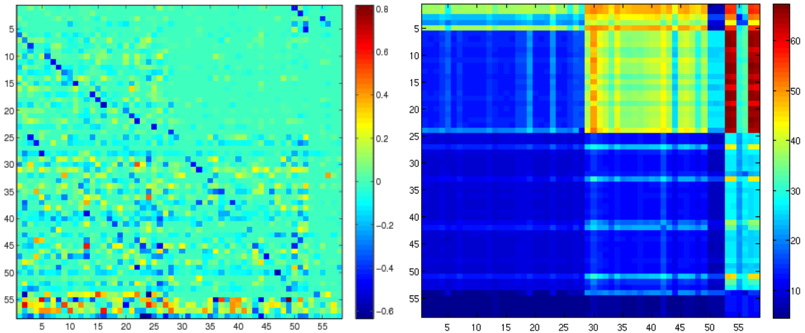
two-stage baseline methods L_1+Biclus, Refit+Biclus, and L_1 row-clus (col-clus) make a significant amount of errors, but still recover moderately accurate clusterings. In contrast, the clusterings by the time-series similarity based methods, BHC-SE and BHC-SE reps, are barely better than random clusterings. To explain this, we first point out that BHC-SE and BHC-SE reps are designed to model time series as noisy observations of deterministic, time-dependent "trends" or "curves" and to group similar curves together, but the time series generated from our stable VAR model all have zero expectation *at all time points* (not just *across time*). As a result, clustering based on similar trends may just be fitting noise in our simulated series. These results on clustering quality suggest that when the underlying cluster structure stems from (Granger) causal relations, clustering methods based on series similarity may give irrelevant results, and we really need methods that explicitly take into account dynamic interaction patterns, such as the one we propose here.

5.2 Modeling T-Cell Activation Gene Expression Time Series

We analyze a gene expression time series dataset[3] collected by [18] from a T-cell activation experiment. To facilitate the analysis, they pre-processed the raw data to obtain 44 replicates of 58 gene time series across 10 unevenly-spaced time points. Recently [3] carried out clustering analysis of these time series data, with their proposed Gaussian process (GP) based Bayesian Hierarchical Clustering (BHC) and quite a few other state-of-the art time series clustering methods. BHC, aided by GP with a cubic spline covariance function, gave the best clustering result as measured by the Biological Homogeneity Index (BHI) [4], which scores a gene cluster based on its number of gene pairs that share certain biological annotations (Gene Ontology terms).

To apply our proposed method, we first normalize each time series to have mean 0 and standard deviation 1 across both time points and replicates, and

[3] Available in the R package longitudinal.



(a) Transition matrix

(b) Average inverse scale L

Fig. 4. Heat maps of the Biclus-EM estimate of A and the inverse scale parameters L averaged over posterior samples; rows and columns permuted according to clusters

then "de-trend" the series by taking the first order difference, resulting in 44 replicates of 58 time series of gene expression differences across 9 time points. We run Algorithm 1 on this de-trended dataset, with all the hyper-parameters and initial values set in the same way as in our simulation study. In 3,000 iterations the algorithm mixes reasonably well; we let it run for another 2,000 iterations and take samples from every 10 iterations, resulting in 200 posterior samples, to compute point estimates for A, cluster indicators $\mathbf{u}$ and $\mathbf{v}$ as described in Sections 4.1 and 4.2. Figures 4(a) and 4(b) show the heat maps of the transition matrix point estimate and the inverse scale parameters λ_{ij}'s averaged over the posterior samples, with rows and columns permuted according to clusters, revealing a quite clear bi-clustering structure.

For competing methods, we use the GP based Bayesian Hierarchical Clustering (BHC) by [3], with two GP covariance functions: cubic spline (BHC-C) and squared-exponential (BHC-SE)[4]. We also apply the two-stage method L_1+Biclus described in our simulation study, but its posterior samples give an average of 15 clusters, which is much more than the number of clusters, around 4, from the spectral analysis described in Section 4.2, suggesting a high level of uncertainty in their posterior inferences about cluster indicators. We thus do not report their results here. The other two simple baselines are: Corr, standing for normalized spectral clustering on the correlation matrix of the 58 time series averaged over all 44 replicates, the number of clusters 2 determined by the spectral gap, and All-in-one, which simply puts all genes in one cluster.

Figure 5 shows the BHI scores[5] given by different methods, and higher-values indicate bettering clusterings. Biclus row and Biclus col respectively denote the

[4] Here we only report results obtained without using replicate information because using replicate information does not give better results. We obtain cluster labels from `http://www.biomedcentral.com/1471-2105/12/399/additional`

[5] We compute BHIs by the BHI function in the R package clValid (version 0.6-4) [1] and the database hgu133plus2.db (version 2.6.3), following [3].

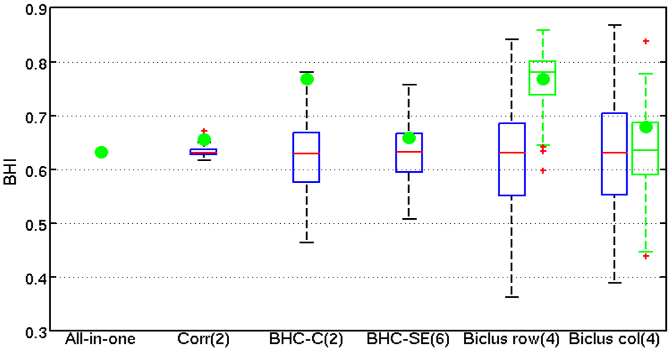
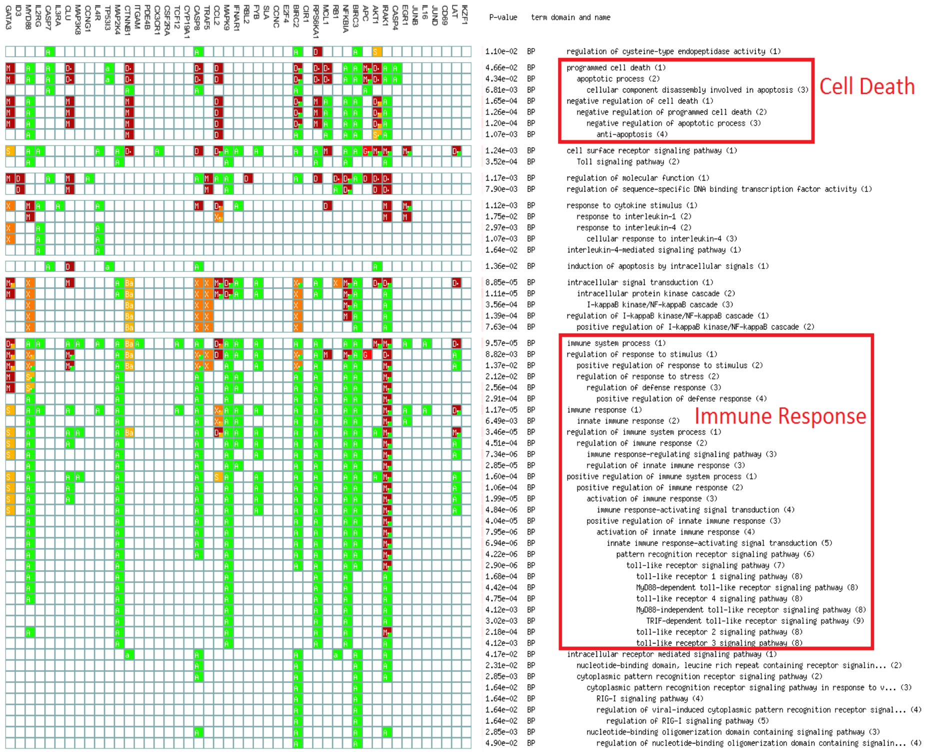


Fig. 5. BHI. Green dots show BHIs of different methods; blue boxes are BHIs obtained by 200 random permutations of cluster labels by those methods; green boxes are BHIs computed on posterior cluster indicator samples from the proposed method. In parentheses are numbers of clusters given by different methods.

row and column clusterings given by our method. To measure the significance of the clusterings, we report BHI scores computed on 200 random permutations of the cluster labels given by each method. For Biclus row and Biclus col, we also report the scores computed on the 200 posterior samples. All-in-one has a BHI score around 0.63, suggesting that nearly two-thirds of all gene pairs share some biological annotations. Corr puts genes into two nearly equal-sized clusters (28 and 30), but does not increase the BHI score much. In contrast, BHC-C and Biclus row achieve substantially higher scores, and both are significantly better than those by random permutations, showing that the improvements are much more likely due to the methods rather than varying numbers or sizes of clusters. We also note that even though Corr and BHC-C both give two clusters, the two BHC-C clusters have very different sizes (48 and 10), which cause a larger variance in their BHI distribution under random label permutations. Lastly, BHC-SE and Biclus col give lower scores that are not significantly better than random permutations. One possible explanation for the difference in scores by Biclus row and Biclus col is that the former bases itself on how genes *affect* one another while the latter on how genes *are affected* by others, and Gene Ontology terms, the biological annotations underlying the BHI function, describe more about genes' active roles or molecular functions in various biological processes than what influence genes.

Finally, to gain more understanding on the clusters by BHC-C and Biclus row, we conduct gene function profiling with the web-based tool **g:Profiler** [19], which performs "statistical enrichment analysis to provide interpretation to user-defined gene lists." We select the following three options: *Significant only*, *Hierarchical sorting*, and *No electronic GO annotations*. For BHC-C, 4 out of 10 genes in the small cluster are found to be associated with the KEGG cell-cycle pathway (04110), but the other 6 genes are not mapped to collectively meaningful annotations. The profiling results of the large BHC-C cluster with 48 genes are in

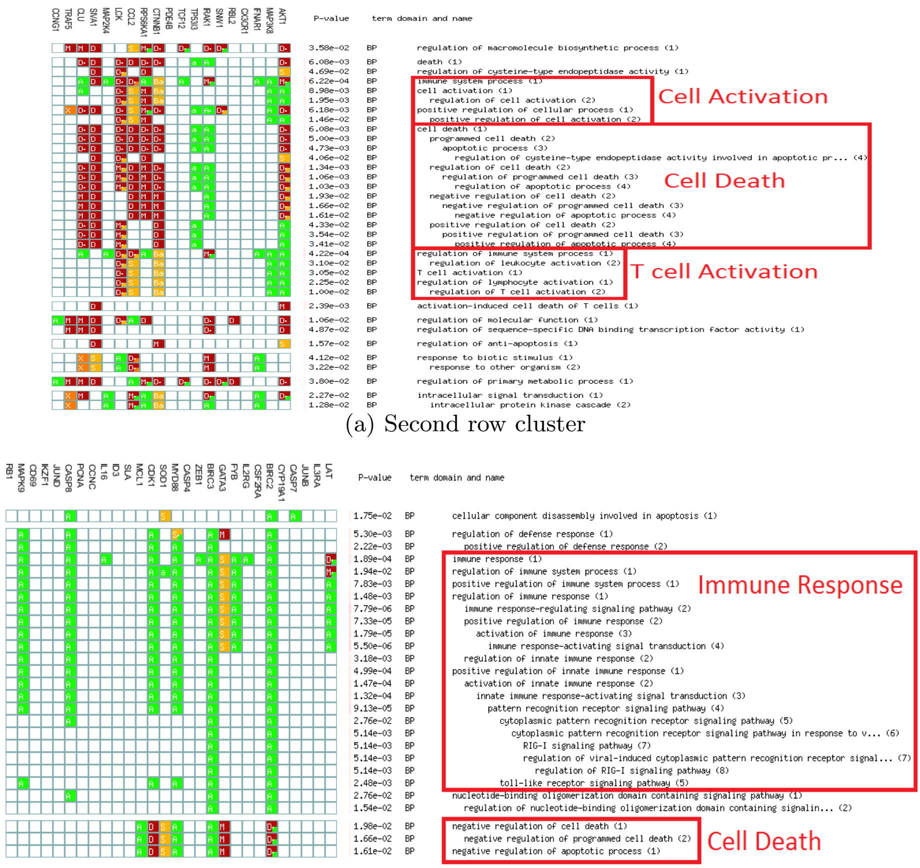


```
P-value   term domain and name

1.10e-02  BP    regulation of cysteine-type endopeptidase activity (1)

4.66e-02  BP    programmed cell death (1)
4.34e-02  BP      apoptotic process (2)
6.81e-03  BP        cellular component disassembly involved in apoptosis (3)
1.65e-04  BP    negative regulation of cell death (1)
1.26e-04  BP      negative regulation of programmed cell death (2)
1.20e-04  BP        negative regulation of apoptotic process (3)
1.07e-03  BP          anti-apoptosis (4)

1.24e-03  BP    cell surface receptor signaling pathway (1)
3.52e-04  BP      Toll signaling pathway (2)

1.17e-03  BP    regulation of molecular function (1)
7.90e-03  BP    regulation of sequence-specific DNA binding transcription factor activity (1)

1.12e-03  BP    response to cytokine stimulus (1)
1.75e-02  BP      response to interleukin-1 (2)
2.97e-03  BP      response to interleukin-4 (2)
1.07e-03  BP        cellular response to interleukin-4 (3)
1.64e-02  BP    interleukin-4-mediated signaling pathway (1)

1.36e-02  BP    induction of apoptosis by intracellular signals (1)

8.85e-05  BP    intracellular signal transduction (1)
1.11e-05  BP      intracellular protein kinase cascade (2)
3.56e-04  BP        I-kappaB kinase/NF-kappaB cascade (3)
1.39e-04  BP    regulation of I-kappaB kinase/NF-kappaB cascade (1)
7.63e-04  BP      positive regulation of I-kappaB kinase/NF-kappaB cascade (2)

9.57e-05  BP    immune system process (1)
8.82e-03  BP    regulation of response to stimulus (1)
1.37e-02  BP      positive regulation of response to stimulus (2)
2.12e-02  BP      regulation of response to stress (2)
2.56e-04  BP        regulation of defense response (3)
2.91e-04  BP          positive regulation of defense response (4)
1.17e-05  BP    immune response (1)
6.49e-03  BP      innate immune response (2)
3.46e-05  BP    regulation of immune system process (1)
4.51e-04  BP      regulation of immune response (2)
7.34e-06  BP        immune response-regulating signaling pathway (3)
2.85e-05  BP        regulation of innate immune response (3)
1.60e-04  BP    positive regulation of immune system process (1)
1.06e-04  BP      positive regulation of immune response (2)
1.99e-05  BP        activation of immune response (3)
4.84e-06  BP          immune response-activating signal transduction (4)
4.04e-05  BP        positive regulation of innate immune response (3)
7.95e-06  BP          activation of innate immune response (4)
6.94e-06  BP            innate immune response-activating signal transduction (5)
4.22e-06  BP              pattern recognition receptor signaling pathway (6)
2.90e-06  BP                toll-like receptor signaling pathway (7)
1.68e-04  BP                  toll-like receptor 1 signaling pathway (8)
4.42e-04  BP                  MyD88-dependent toll-like receptor signaling pathway (8)
4.75e-04  BP                  toll-like receptor 4 signaling pathway (8)
4.12e-03  BP                  MyD88-independent toll-like receptor signaling pathway (8)
3.02e-03  BP                    TRIF-dependent toll-like receptor signaling pathway (9)
2.18e-04  BP                  toll-like receptor 2 signaling pathway (8)
4.12e-03  BP                  toll-like receptor 3 signaling pathway (8)
4.17e-02  BP    intracellular receptor mediated signaling pathway (1)
2.31e-02  BP      nucleotide-binding domain, leucine rich repeat containing receptor signalin... (2)
2.85e-03  BP      cytoplasmic pattern recognition receptor signaling pathway (2)
1.64e-02  BP        cytoplasmic pattern recognition receptor signaling pathway in response to v... (3)
1.64e-02  BP          RIG-I signaling pathway (4)
1.64e-02  BP          regulation of viral-induced cytoplasmic pattern recognition receptor signal... (4)
1.64e-02  BP            regulation of RIG-I signaling pathway (5)
2.85e-03  BP        nucleotide-binding oligomerization domain containing signaling pathway (3)
4.90e-02  BP          regulation of nucleotide-binding oligomerization domain containing signalin... (4)
```

Fig. 6. Gene functional profiling of the large BHC-C cluster

Figure 6; for better visibility we show only the Gene Ontology (GO) terms and high-light similar terms with red rectangles and tags. About a half of the terms are related to cell death and immune response, and the other half are lower-level descriptions involving, for example, signaling pathways. For Biclus row, we report the profiling results of only the two larger clusters (the second and the third) in Figure 7, because the two smaller clusters, each containing 5 genes, are not mapped to collectively meaningful GO terms. Interestingly, the two large Biclus row clusters are associated with T-cell activation and immune response respectively, and together they cover 41 of the 48 genes in the large BHC-C cluster. This suggests that our method roughly splits the large BHC-C cluster into two smaller ones, each being mapped to a more focused set of biological annotations. Moreover, these Biclus profiling results, the heat map in Figure 4(a), and the contingency table (shown in the right) between the row and column clusters altogether constitute a nice resonance with the fact that T-cell activation results from, rather than leading to, the emergence of immune responses.

col row	1	2	3	4
1	0	0	3	2
2	17	2	0	0
3	10	17	0	2
4	1	2	0	2



P-value	term domain and name	
3.58e-02	BP	regulation of macromolecule biosynthetic process (1)
6.08e-03	BP	death (1)
4.69e-02	BP	regulation of cysteine-type endopeptidase activity (1)
6.22e-04	BP	immune system process (1)
8.98e-03	BP	cell activation (1)
1.95e-03	BP	regulation of cell activation (2)
6.18e-03	BP	positive regulation of cellular process (1)
1.46e-02	BP	positive regulation of cell activation (2)
6.08e-03	BP	cell death (1)
5.00e-03	BP	programmed cell death (2)
4.73e-03	BP	apoptotic process (3)
4.06e-02	BP	regulation of cysteine-type endopeptidase activity involved in apoptotic pr... (4)
1.34e-03	BP	regulation of cell death (2)
1.06e-03	BP	regulation of programmed cell death (3)
1.03e-03	BP	regulation of apoptotic process (4)
1.93e-02	BP	negative regulation of cell death (2)
1.66e-02	BP	negative regulation of programmed cell death (3)
1.61e-02	BP	negative regulation of apoptotic process (4)
4.33e-02	BP	positive regulation of cell death (2)
3.54e-02	BP	positive regulation of programmed cell death (3)
3.41e-02	BP	positive regulation of apoptotic process (4)
4.22e-04	BP	regulation of immune system process (1)
3.10e-02	BP	regulation of leukocyte activation (2)
3.05e-02	BP	T cell activation (1)
2.25e-02	BP	regulation of lymphocyte activation (1)
1.00e-02	BP	regulation of T cell activation (2)
2.39e-03	BP	activation-induced cell death of T cells (1)
1.06e-02	BP	regulation of molecular function (1)
4.87e-02	BP	regulation of sequence-specific DNA binding transcription factor activity (1)
1.57e-02	BP	regulation of anti-apoptosis (1)
4.12e-02	BP	response to biotic stimulus (1)
3.22e-02	BP	response to other organism (2)
3.80e-02	BP	regulation of primary metabolic process (1)
2.27e-02	BP	intracellular signal transduction (1)
1.28e-02	BP	intracellular protein kinase cascade (2)

(a) Second row cluster

P-value	term domain and name	
1.75e-02	BP	cellular component disassembly involved in apoptosis (1)
5.30e-03	BP	regulation of defense response (1)
2.22e-03	BP	positive regulation of defense response (2)
1.89e-04	BP	immune response (1)
1.94e-02	BP	regulation of immune system process (1)
7.83e-03	BP	positive regulation of immune system process (1)
1.48e-03	BP	regulation of immune response (1)
7.79e-06	BP	immune response-regulating signaling pathway (2)
7.33e-05	BP	positive regulation of immune response (2)
1.79e-05	BP	activation of immune response (3)
5.50e-06	BP	immune response-activating signal transduction (4)
3.18e-03	BP	regulation of innate immune response (2)
4.99e-04	BP	positive regulation of innate immune response (1)
1.47e-04	BP	activation of innate immune response (2)
1.32e-04	BP	innate immune response-activating signal transduction (3)
9.13e-05	BP	pattern recognition receptor signaling pathway (4)
2.76e-02	BP	cytoplasmic pattern recognition receptor signaling pathway (5)
5.14e-03	BP	cytoplasmic pattern recognition receptor signaling pathway in response to v... (6)
5.14e-03	BP	RIG-I signaling pathway (7)
5.14e-03	BP	regulation of viral-induced cytoplasmic pattern recognition receptor signal... (7)
5.14e-03	BP	regulation of RIG-I signaling pathway (8)
2.48e-03	BP	toll-like receptor signaling pathway (5)
2.76e-02	BP	nucleotide-binding oligomerization domain containing signaling pathway (1)
1.54e-02	BP	regulation of nucleotide-binding oligomerization domain containing signalin... (2)
1.98e-02	BP	negative regulation of cell death (1)
1.66e-02	BP	negative regulation of programmed cell death (2)
1.61e-02	BP	negative regulation of apoptotic process (1)

(b) Third row cluster

Fig. 7. Gene functional profiling of two large row clusters by the proposed method

6 Conclusion

We develop a nonparametric Bayesian method to simultaneously infer sparse VAR models and bi-clusterings from multivariate time series data, and demonstrate its effectiveness via simulations and experiments on real T-cell activation gene expression time series, on which the proposed method finds a more biologically interpretable clustering than those by some state-of-the art methods. Future directions include modeling signs of transition matrix entries, generalizations to higher-order VAR models, and applications to other real time series.

References

1. Brock, G., Pihur, V., Datta, S., Datta, S.: clvalid: An R package for cluster validation. Journal of Statistical Software 25(4), 1–22 (2008)
2. Busygin, S., Prokopyev, O., Pardalos, P.: Biclustering in data mining. Computers & Operations Research 35(9), 2964–2987 (2008)

3. Cooke, E., Savage, R., Kirk, P., Darkins, R., Wild, D.: Bayesian hierarchical clustering for microarray time series data with replicates and outlier measurements. BMC Bioinformatics 12(1), 399 (2011)
4. Datta, S., Datta, S.: Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. BMC Bioinformatics 7(1), 397 (2006)
5. Fujita, A., Sato, J., Garay-Malpartida, H., Yamaguchi, R., Miyano, S., Sogayar, M., Ferreira, C.: Modeling gene expression regulatory networks with the sparse vector autoregressive model. BMC Systems Biology 1(1), 39 (2007)
6. Girvan, M., Newman, M.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99(12), 7821 (2002)
7. Granger, C.: Investigating causal relations by econometric models and cross-spectral methods. Econometrica, 424–438 (1969)
8. Heller, K., Ghahramani, Z.: Bayesian hierarchical clustering. In: The 22nd International Conference on Machine Learning, pp. 297–304. ACM (2005)
9. Herman, I., Melançon, G., Marshall, M.: Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics 6(1), 24–43 (2000)
10. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification 2(1), 193–218 (1985)
11. Lozano, A., Abe, N., Liu, Y., Rosset, S.: Grouped graphical granger modeling for gene expression regulatory networks discovery. Bioinformatics 25(12), i110 (2009)
12. Marlin, B.M., Schmidt, M., Murphy, K.P.: Group sparse priors for covariance estimation. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009), Montreal, Canada (2009)
13. Meeds, E., Roweis, S.: Nonparametric Bayesian biclustering. Technical report, Department of Computer Science, University of Toronto (2007)
14. Mills, T.C.: The Econometric Modelling of Financial Time Series, 2nd edn. Cambridge University Press (1999)
15. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Advances in Neural Information Processing Systems (2001)
16. Porteous, I., Bart, E., Welling, M.: Multi-hdp: A non-parametric bayesian model for tensor factorization. In: Proc. of the 23rd National Conf. on Artificial Intelligence, pp. 1487–1490 (2008)
17. Ramoni, M., Sebastiani, P., Kohane, I.: Cluster analysis of gene expression dynamics. Proceedings of the National Academy of Sciences 99(14), 9121 (2002)
18. Rangel, C., Angus, J., Ghahramani, Z., Lioumi, M., Sotheran, E., Gaiba, A., Wild, D., Falciani, F.: Modeling T-cell activation using gene expression profiling and state-space models. Bioinformatics 20(9), 1361–1372 (2004)
19. Reimand, J., Arak, T., Vilo, J.: g: Profiler – a web server for functional interpretation of gene lists (2011 update). Nucleic Acids Research 39(suppl. 2), W307–W315 (2011)
20. Schaeffer, S.: Graph clustering. Computer Science Review 1(1), 27–64 (2007)
21. Sethuraman, J.: A constructive definition of Dirichlet priors. Statistica Sinica 4, 639–650 (1994)
22. Shojaie, A., Basu, S., Michailidis, G.: Adaptive thresholding for reconstructing regulatory networks from time-course gene expression data. Statistics in Biosciences, 1–18 (2011)
23. Tsay, R.S.: Analysis of financial time series. Wiley-Interscience (2005)
24. Zou, H.: The adaptive lasso and its oracle properties. Journal of the American Statistical Association 101(476), 1418–1429 (2006)

Discriminative Factor Alignment across Heterogeneous Feature Space

Fangwei Hu[1], Tianqi Chen[1], Nathan N. Liu[2], Qiang Yang[2], and Yong Yu[1]

[1] Shanghai Jiao Tong University, Shanghai, China
{hufangwei,tqchen,yyu}@apex.sjtu.edu.cn
[2] Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong*
{nliu,qyang}@cse.ust.hk

Abstract. Transfer learning as a new machine learning paradigm has gained increasing attention lately. In situations where the training data in a target domain are not sufficient to learn predictive models effectively, transfer learning leverages auxiliary source data from related domains for learning. While most of the existing works in this area are only focused on using the source data with the same representational structure as the target data, in this paper, we push this boundary further by extending transfer between text and images.

We integrate documents , tags and images to build a heterogeneous transfer learning factor alignment model and apply it to improve the performance of tag recommendation. Many algorithms for tag recommendation have been proposed, but many of them have problem; the algorithm may not perform well under cold start conditions or for items from the long tail of the tag frequency distribution. However, with the help of documents, our algorithm handles these problems and generally outperforms other tag recommendation methods, especially the non-transfer factor alignment model.

1 Introduction

Tag recommendation has found many applications ranging from personal photo albums to multimedia information delivery. In the past, tag recommendation has met two major difficulties. First, the annotated images for training are often in short supply, and annotating new images involves much human labor. Hence, annotated training data is often sparse, and further tags included in training data may be from the long tail of the frequency distribution. Second, words usually have synonym; e.g. two words may have same or similar meaning. We would like to find the latent links between these words. How to effectively overcome these difficulties and build a good tag recommendation system therefore becomes a challenging research problem. While annotated images are expensive, abundant text data are easier to obtain. This motivates us to find way to use the readily available text data to help improve the tag recommendation performance.

In the past, several approaches have been proposed to solve the 'lack of data' problem. Recently, transfer learning methods [1] have been proposed to use knowledge from

* Part of this work was done while the first author was a visiting student in Hong Kong University of Science Technology.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 757–772, 2012.

auxiliary data in a different but related domain to help learn the target tasks. However, a commonality among most transfer learning methods so far is that the data from different domains have the same feature space.

In some scenarios, given a target task, one may easily collect much auxiliary data that are represented in a different feature space. For example, our task is to recommend tags for an image with a tiger in it. We have only a few annotated images for training. And we can easily collect a large number of text documents from the Web, e.g. Wikipedia. In this case, we can model the tag recommendation task as the target task, where we have some annotated data and some auxiliary data. In the target domain, the data are represented in pixels. Also in our case, the auxiliary domain, or the source domain, is the text domain, which contains text documents. Now, what we care about is whether it is possible to use the cheap auxiliary data to help improve the performance of the tag recommendation task. This is an interesting and difficult question, since the relationship between text and images is not explicitly given. This problem has been referred to as a *Heterogeneous Transfer Learning* problem [2]. In this paper, we focus on heterogeneous transfer learning for tag recommendation by exploring knowledge transfer from auxiliary text data.

In tag recommendation, a key issue for us to address is to discover a new and improved common representation for both images and tags to boost the recommendation performance. In this paper, we investigate how to obtain a reasonable common feature space from both annotated images and auxiliary text data. Although images and text are represented in different feature spaces, we can transfer knowledge from text to images via tags which are related both to images and text. We propose a factor alignment model to discover the common feature space and modify it into a heterogeneous transfer learning factor alignment model, which can handle the auxiliary data effectively. A common space is then learned to better calculate the metric between an image and a tag. We illustrate the overall framework in Figure 1. Compared to self-taught learning [3], our approach can use a different feature representation (i.e., text) for transfer learning. Compared to translated learning [4] and Zhu et al. [5], their tasks are different from ours and our approach does not need to compute the total correlation between image feature and word features.

2 Related Work

2.1 Image Annotation

A closely related area is image annotation. Duygulu et al. [6] regarded image annotation as a machine translating process. Some other researchers model the joint probability of images regions and annotations. Barnard et al. [7] investigated image annotation under probabilistic framework and put forward a number of models for the joint distribution of image blobs and words. Blei et al. [8] developed *correspondence latent Dirichlet allocation* to model the joint distribution. In Lavrenko et al. [9], the *continuous-space relevance model* was proposed to better handle continuous feature and be free from the influence of image blob clustering. In Carneiro et al. [10], image annotation is posed

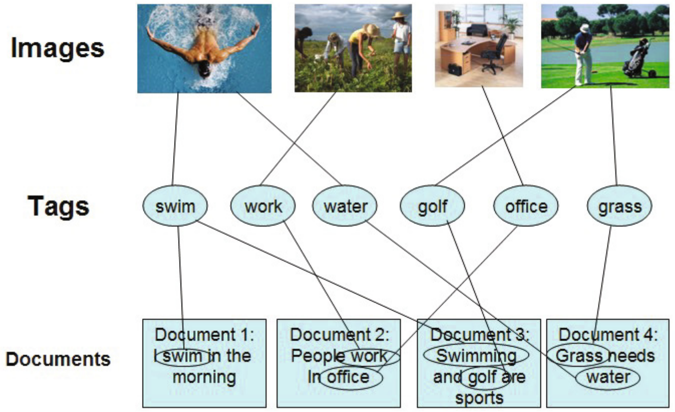

Images
Tags
swim
work
water
golf
office
grass
Documents
Document 1: I swim in the morning
Document 2: People work In office
Document 3: Swimming and golf are sports
Document 4: Grass needs water

Fig. 1. Source data used for our transfer learning algorithms. Our proposed *heterogeneous transfer learning for tag recommendation* takes all three information, i.e. images, tags and documents as inputs.

as classification problems where each class is defined by images sharing a common semantic label. In this paper, we use an image annotation algorithm proposed in Makadia et al. [11] to solve the problem of tag recommendation as a baseline.

2.2 Image Retagging

There are some efforts on improving unreliable descriptive keywords of images. They focus on imprecise annotation refinement, i.e., identifying and eliminating the imprecise annotation keywords produced by the automatic image annotation algorithms. As a pioneering work, Jin et al. [12] used WordNet to estimate the semantic correlation among the annotated keywords and remove the weakly-correlated ones. However, this method can only achieve limited success since it totally ignores the visual content of the images. To address this problem, Wang et al. [13] proposed a content-based approach to re-rank the automatically annotated keywords of an image and only reserve the top ones as the refined results and Liu et al. [14] proposed to refine the tags based on the visual and semantic consistency residing in the social images, which assigns similar tags to visually similar images. Later, Liu et al. [15] formulated this retagging process as a multiple graph-based multi-label learning problem, which simultaneously explores the visual content of the images, semantic correlation of the tags and the prior information provided by users.

2.3 Visual Contextual Advertising

Another closely related area is visual contextual advertising, which aims to recommend advertisements for Web images without the help of any textual context, such as surrounding text for the images. In Chen et al. [16], they exploit the annotated image data

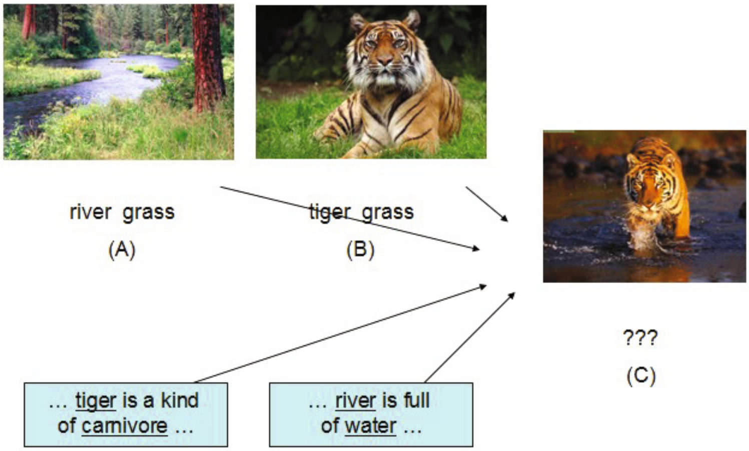

river grass
(A)
tiger grass
(B)
???
(C)
... tiger is a kind of carnivore ...
... river is full of water ...

Fig. 2. A case in which document may help the tag recommendation

from social Web sites such as Flickr to link the visual feature space and the word space. To be specific, they present a unified generative model, ViCAD, to deal with visual contextual advertising. ViCAD runs in several steps. First, they model the visual contextual advertising problem with a Markov chain which utilizes annotated images to transform images from the image feature space to the word space. With the representations of images in word space, a language model for information retrieval is then applied to find the most relevant advertisements. If we regard tag as one-word advertisement, ViCAD can handle the problem of tag recommendation. Hence we use ViCAD as one of our baselines.

3 Motivation

Before describing our proposed method in detail we first illustrate a motivating example showing how text data help improve the tag recommendation performance for images. As shown in Figure 2, we may have image A and image B as training data and image C as testing data. As mentioned before, we can't recommend other tags except 'river' and 'tiger' for image C. However, by using some additional auxiliary text documents where tags co-occur frequently, we may establish a strong similarity between tags. If we have a document including '*... a tiger is a kind of carnivore ...*', we will build a similarity between 'tiger' and 'carnivore', which may cause the recommendation system to recommend 'carnivore' for image C. And if we have a document including '*... a river is full of water ...*', 'river' and 'water' will also be regarded as being related. Consequently, 'water' may be also recommended; this can reduce the data sparsity in the image domain and word domain.

4 Algorithm

4.1 Problem Formulation

First we define the problem of our tag recommendation task formally. Suppose we are given annotated data instances $X = \{z_i, t_i\}_{i=1}^{n}$ and some test images $X^* = \{z_i^*, t_i^*\}_{i=n+1}^{n+m}$, where $z_i \in \mathbb{R}^d$ is an input vector of image features and $t_i \in \mathbb{R}^h$ is the corresponding tags of image i, where h is the number of tags. For example, if an image z_i is annotated by tags α and β with $\alpha, \beta \in \{1, \ldots, h\}$, then $t_i = [0, \ldots, 1, \ldots, 1, \ldots, 0]$ is a vector of dimensionality h with all zeros but one's in the α and β positions. Using "bag-of-words" [17] to represent image features, we can assume that the feature values are nonnegative. n and m are the numbers of training and testing instances, respectively. In addition, we also have a set of auxiliary text documents $F = \{f_i\}_{i=1}^{k}$, $f_i \in \mathbb{R}^s$ is a document represented by a vector of bag-of-words, and k is the number of auxiliary documents. Here, we notice that s doesn't have to be equal to h. Actually set of tags for words is a subset of the set for documents, which means $h \leq s$. Our goal is to learn a function $g(\cdot, \cdot)$ from X and F, that can estimate the correlation between a given image and a tag as accurately as possible. Applying $g(\cdot, \cdot)$ on X^*, we can rank the correlation to obtain the recommended tag list for test images. We summarize the problem definition in Table 1. For convenience, we denote $Z = \{z_i\}_{i=1}^{l} \in \mathbb{R}^{l \times d}$ and $T = \{t_i\}_{i=1}^{l} \in \mathbb{R}^{l \times h}$ the image features and text tags of images separately. Furthermore, we abuse the notation X, X^*, Z, and T to represent the data matrices with instances x_i, x_i^*, z_i, and t_i being row vectors in them.

Table 1. Problem formulation

Learning objective	Make predictions on target test images
Target tag recommendation	Training images: $X = \{z_i, t_i\}_{i=1}^{n}$ Testing images: $X^* = \{z_i^*, t_i^*\}_{i=n+1}^{n+m}$
Auxiliary source data	Text documents: $F = \{f_i\}_{i=1}^{k}$

4.2 Algorithm Description

Given a set of images $Z \in \mathbb{R}^{l \times d}$ with their corresponding tags $T \in \mathbb{R}^{l \times h}$, and a set of documents $F \in \mathbb{R}^{k \times s}$, we wish build a connection between images and text documents. Each image can be annotated by tags, and some images may share one or multiple tags. If two images are annotated by shared tags, they tend to be related to each other semantically. Similarly, if two tags co-occur in annotations of shared images, they tend to be related to each other. This image-tag bipartite graph is represented via the tag matrix T. If a tag, more precisely, the text word of the tag, occurs in a document, then there is an edge connecting the tag and the document. We use a matrix $F \in \mathbb{R}^{k \times s}$ to represent the document-tag bipartite graph, where $F_{ij} = n$ if the j^{th} tag appears n times in the i^{th} document.

Image-Tag Ranking Model. In this section, we would like to first give the ranking model only leveraging the information of annotated images. Our intuitive idea is to project the image feature space and tag feature space to a common latent space, and then we can use dot product to calculate the correlation between an image and a tag. Hence we define $W \in \mathbb{R}^{d \times g}$ as a projection matrix to project the image feature space to the latent space and we also define $P \in \mathbb{R}^{h \times g}$as the latent space matrix of tags (i.e. $\mathbf{p}_i \in \mathbb{R}^g$ is the latent feature vector of tag i). Now for any image i and any tag j, we first project $\mathbf{z}_i \in \mathbb{R}^d$ to the common space as

$$\mathbf{c}_i = \mathbf{z}_i W \in \mathbb{R}^g \tag{1}$$

We can calculate the correlation between image i and tag j as

$$\hat{T}_{ij} = \mathbf{c}_i \mathbf{p}_j^\top = \mathbf{z}_i W \mathbf{p}_j^\top \tag{2}$$

Using matrix format to represent the formula, we obtain the equation

$$\hat{T} = ZWP^\top \tag{3}$$

Now we would like to define a loss function to measure the rank distance of the predicting matrix and real training matrix. Inspired by the metric of area under the ROC curve (AUC) [18], we define the loss function as

$$L(T, \hat{T}) = -\sum_{u,i,j} r_{ij}^{(u)} \ln \hat{r}_{ij}^{(u)} + (1 - r_{ij}^{(u)}) \ln(1 - \hat{r}_{ij}^{(u)}) \tag{4}$$

where

$$r_{ij}^{(u)} = \begin{cases} 1 \text{ }, T_{ui} - T_{uj} > 0 \\ 0 \text{ }, otherwise \end{cases} \tag{5}$$

$$\hat{r}_{ij}^{(u)} = \frac{1}{1 + e^{-(\hat{T}_{ui} - \hat{T}_{uj})}} \tag{6}$$

We notice that

$$r_{ij}^{(u)} \ln \hat{r}_{ij}^{(u)} + (1 - r_{ij}^{(u)}) \ln(1 - \hat{r}_{ij}^{(u)}) \tag{7}$$

is comfortingly symmetric (swapping i and j should leave the result invariant). In order to simplify the formula, we define

$$D(T) = \{(u, i, j) | T_{ui} - T_{uj} > 0\} \tag{8}$$

Now we can rewrite the loss function as

$$L(T, \hat{T}) = -\sum_{u,i,j \in D(T)} r_{ij}^{(u)} \ln \hat{r}_{ij}^{(u)} + (1 - r_{ij}^{(u)}) \ln(1 - \hat{r}_{ij}^{(u)}) \tag{9}$$

$$= -\sum_{u,i,j \in D(T)} 1 \cdot \ln \hat{r}_{ij}^{(u)} + (1 - 1) \ln(1 - \hat{r}_{ij}^{(u)}) \tag{10}$$

$$= -\sum_{u,i,j \in D(T)} \ln \hat{r}_{ij}^{(u)} \tag{11}$$

So far, our corresponding optimization problem becomes

$$\min_{W,P} L(T, \hat{T}) + \lambda_1 \|W\|_F^2 + \lambda_2 \|P\|_F^2 \tag{12}$$

where λ_1, λ_2 are nonnegative parameters to control the responding regularization terms, and $\|\cdot\|_F$ is the Frobenius norm. In next section, we will apply this model on text documents.

Doc-Tag Ranking Model. Now let us consider the information included by text documents, and we would like to apply the ranking model mentioned in the last section to text documents. Hence the problem in this section is to recommend tags for each text document. Similarly, we define $Q \in \mathbb{R}^{h \times g}$ as the latent space matrix of tags. However, unlike images, we directly define $B \in \mathbb{R}^{s \times g}$ as the latent space matrix of documents. Now for any document i and any tag j, we can obtain the correlation between document i and tag j as

$$\hat{F}_{ij} = \mathbf{b}_i \mathbf{q}_j^\top \tag{13}$$

The matrix format of the calculation is

$$\hat{F} = BQ^\top \tag{14}$$

Therefore, the corresponding optimization of this problem is

$$\min_{B,Q} Loss(F, \hat{F}) + \lambda_3 \|B\|_F^2 + \lambda_4 \|Q\|_F^2 \tag{15}$$

where λ_3, λ_4 are small nonnegative numbers, $\lambda_3 \|B\|_F^2$ and $\lambda_4 \|Q\|_F^2$ serve as a regularization term to improve the robustness.

Joint Ranking Model. There are many ways to transfer knowledge of text data to image data. Our idea in this paper is not to calculate the correlation between image features and text features, but rather to transfer via the latent space of tags. By forcing P and Q to be approximate, the ranking model only for images can incorporate the information of text data. Compared to forcing two latent matrices (i.e. P and Q) to be absolutely the same, our model uses soft constraints leveraging λ_0 to control the similarity of these two matrices, which is more flexible. $\lambda_0 = 0$ implies that P and Q are uncorrelated and documents do nothing to help learning. Then $\lambda_0 = \infty$ denotes that P should be equal to Q, which becomes a hard constraint. To achieve this goal, we solve the following problem,

$$\min_{W,B,P,Q} L(T, \hat{T}) + L(F, \hat{F}) + \lambda_0 \|P - Q\|_F^2 + R(W, B, P, Q) \tag{16}$$

where $R(W, B, P, Q)$ is the regularization function to control the complexity of the latent matrices W, B, P and Q, and λ_0 controls the strength to constrain the similarity of P and Q. As mentioned in previous sections, we define the regularization function as

$$R(W, B, P, Q) = \lambda_1 \|W\|_F^2 + \lambda_2 \|P\|_F^2 + \lambda_3 \|B\|_F^2 + \lambda_4 \|Q\|_F^2 \tag{17}$$

where λ_1, λ_2, λ_3 and λ_4 are nonnegative parameters to control the responding regularization terms. In this paper, we set $\lambda_1 = \lambda_3 = 0.004$ and $\lambda_2 = \lambda_4 = 0.006$ using cross validation. Equation 16 is the joined ranking model of T and F with regularization. The bridge of transfer P and Q are ensured to capture both the structures of image matrix T and the document matrix F. Once we find the optimal W and P and project test images to latent space, we can apply the dot product to estimate the rating for tags given by images.

Update Rule. Equation 16 can be solved using gradient methods, such as the stochastic gradient descent and quasi-Newton methods. In this paper, we use stochastic gradient descent. Then the main computation of the gradient method gives the update rule for all variables. For $(u, i, j) \in D(T)$,

$$\mathbf{p}_k = \mathbf{p}_k + \eta(\hat{e}_T(u,i,j)Z_{uk}(\mathbf{p}_i - \mathbf{p}_j) - \lambda_1 \mathbf{p}_k) \tag{18}$$

$$\mathbf{p}_k = \mathbf{p}_k + \eta(\hat{e}_T(u,i,j)\gamma_{ij}(k)(\mathbf{z}_u W) - \lambda_2 \mathbf{p}_k - \lambda_0(\mathbf{p}_k - \mathbf{q}_k)) \tag{19}$$

where

$$\hat{e}_T(u,i,j) = 1 - \frac{1}{1 + e^{-(\hat{T}_{ui} - \hat{T}_{uj})}} \tag{20}$$

and

$$\gamma_{ij}(k) = \begin{cases} 1 & , k = i \\ -1 & , k = j \\ 0 & , otherwise \end{cases} \tag{21}$$

Then for $(u, i, j) \in D(F)$,

$$\mathbf{b}_k = \mathbf{b}_k + \eta(\hat{e}_F(u,i,j)\sigma_u(k)(\mathbf{q}_i - \mathbf{q}_j) - \lambda_3 \mathbf{b}_k) \tag{22}$$

$$\mathbf{q}_k = \mathbf{q}_k + \eta(\hat{e}_F(u,i,j)\gamma_{ij}(k)\mathbf{b}_u - \lambda_4 \mathbf{q}_k - \lambda_0(\mathbf{p}_k - \mathbf{q}_k)) \tag{23}$$

Here

$$\hat{e}_F(u,i,j) = 1 - \frac{1}{1 + e^{-(\hat{F}_{ui} - \hat{F}_{uj})}} \tag{24}$$

and

$$\sigma_u(k) = \begin{cases} 1 & , k = u \\ 0 & , otherwise \end{cases} \tag{25}$$

Prediction. Now we will present the process of prediction of test image set. For any image $\mathbf{x}_i^*$ in test set X^*, we first project $\mathbf{z}_i^*$ to the common space as

$$\mathbf{d}_i = \mathbf{z}_i^* W \in \mathbb{R}^g \tag{26}$$

Therefore, the definition of $f(x_i^*, tag_j)$ is as

$$g(x_i^*, tag_j) = \mathbf{d}_i \mathbf{p}_j^\top = \mathbf{z}_i^* W \mathbf{p}_j^\top \tag{27}$$

Using matrix format to represent the prediction matrix, we obtain the equation

$$\hat{R} = Z^* W P^\top \tag{28}$$

Notice that $\hat{R} \in \mathbb{R}^{m \times h}$ can help us rank the tags for testing images. For any testing image u, if $\hat{R}_{ui} > \hat{R}_{uj}$, we can consider tag i is more related to the image than tag j. Thus we will get the recommended tag list in order by ranking the rating. Putting it together, our overall heterogeneous transfer learning algorithm is referred to as **HTLFA**, which stands for Heterogeneous Transfer Learning for Factor Alignment.

5 Experiments

Our experiments are designed to demonstrate the effectiveness of exploiting text in our heterogeneous learning algorithm.

5.1 Dataset and Processing

We use annotated images from Flickr crawled during December 2009. We collected 528,587 images and 20,000 distinct related tags. Each of these tags is a single word. We crawled text from Wikipedia as our auxiliary data, from which we picked out 539,460 documents. Each of the documents includes more than 100 tag words mentioned above.

Data preprocessing is applied to the raw data. We use the "bag-of-words" model [17] to represent each image. First interesting points were detected and described by SIFT descriptors [19]. Then we cluster a random subset of all interesting points to obtain a codebook. Similar to Sivic et al. [20], we set the number of clusters to be 1,000. Using this codebook, each image is converted into a vector for further tag-recommendation uses. One image at most has 208 tags and at least has 1 tag. On average each image has about 9 tags. And for documents, we also converted them into 30,000-dim vectors using 30,000 tag words. The extra 10000 tags are selected from popular tags included in documents which do not appear in origin 20000 tags.

Actually, we don't regard all tags not related to images (or documents) as negative samples. For an image (or document), we just sample n' tags from unrelated tags as negative sample and the other tags are regarded as neutral items. Here n' is proportional to the size of the positive set. For example, if image A has 10 annotated tags, then we randomly select $n' = 10k'$ tags from unrelated ones as negative sample. Here k' is a proportion parameter. At last, we set latent space dimension $g = 100$.

5.2 Evaluation and Baseline Methods

As for evaluation metrics, we choose the precision at n (or **P**@n), which is the portion of related tags in the topmost n recommendations to evaluate the experimental results. In order to make the results more solid, we also use **MAP** to measure the precision. To obtain ground truth results, we reserve about a quarter of all images from the Flickr image set as test set and regard the annotated tags as ground truth. Before we formulate these two evaluation metrics, we first define $M(i)$ as the number of annotated tags belonging to test image i and $loc_i(j)$ as the position of tag j in the recommendation list for test image i. Then for any test image i, we rank the loc_i by ascending order. Now we can present the expression of **P**@n,

$$\mathbf{P}@\mathrm{n} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{M(i)} \frac{\psi(loc_i(j)<n)}{n}}{m} \tag{29}$$

where m is the number of test images and $\psi(Con)$ is a condition function,

$$\psi(Con) = \begin{cases} 1 \text{ , } Con \text{ holds} \\ 0 \text{ , } otherwise \end{cases} \tag{30}$$

Then the expression of **MAP** is formulated as:

$$\mathbf{MAP} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{M(i)}\frac{j}{loc_i(j)}}{m} \tag{31}$$

We compare our proposed method with three baselines for tag recommendation. The three baselines and our proposed method are summarized as follows,

- **NN** This baseline is reported in Makadia et al. [11]. This baseline annotation methods are comprised of an image distance measure for nearest neighbor ranking, combined with a label transfer measure. In this paper, we measure the distance using SIFT feature vector of images.
- **ViCAD** We implemented the method proposed in Chen et al. [16] as another baseline, which directly transforms images from a image feature space to a word space utilizing the knowledge from images with annotations from Flickr. Then each dimension of the word space means the correlation between the image and a word. Since tags can be regarded as single word advertisements, we can directly get the recommendation list of tags by ranking the correlation.
- **FA** The name of this baseline is short for factor alignment, which is a non-transfer algorithm corresponding to the loss function 12. In this baseline, we do not use the documents to help learning the projection matrix. Without the transformation of knowledge from documents to images, we simply train a ranking model only on the image-tag matrix. As mentioned above, we apply an image-tag ranking model on training data to discover the common representation.
- **HTLFA** This denoted our proposed method, which uses all the auxiliary data i.e. documents. The parameter settings are discussed in the following section.

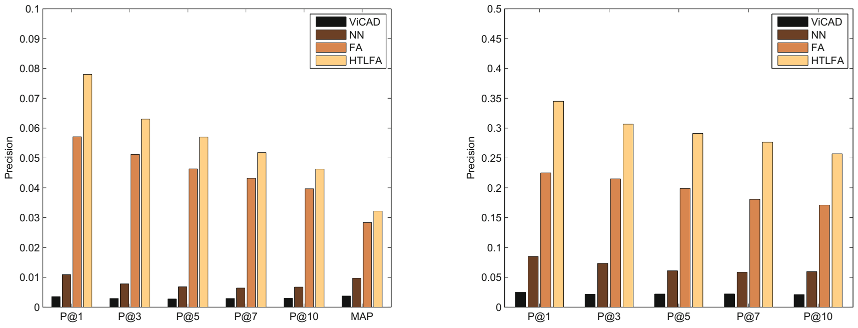
Table 2. Comparison with baselines

	ViCAD	**NN**	**FA**	**HTLFA**
P@1	0.003500	0.010900	0.057100	0.077990
P@2	0.003050	0.009200	0.054250	0.069550
P@3	0.002900	0.007800	0.051200	0.063033
P@4	0.002750	0.006975	0.048975	0.059275
P@5	0.002760	0.006820	0.046340	0.057000
P@6	0.002900	0.006550	0.044850	0.053933
P@7	0.002886	0.006400	0.043200	0.051800
P@8	0.002925	0.006513	0.041712	0.049675
P@9	0.002922	0.006511	0.040778	0.048000
P@10	0.002970	0.006740	0.039670	0.046270
MAP	0.003744	0.009711	0.028368	0.032200

5.3 Comparison with Baselines

In the first experiment, we compare our method with three baselines on the same tag recommendation task. We randomly select 10,000 images from the test set as test data. The **P**@n results with respect to **NN**, **FA**, **ViCAD** and our model are given in Figure 3(a) and Table 2. In this experiment, for **HTLFA**, we set the parameter λ_0 in Equation 16 to 0.05. As we can see from Figure 3(a), our proposed **HTLFA**, which use documents to help learn projection matrices for rating, outperforms other baselines, especially it is better than **FA**. **NN** and **ViCAD** which are generative models without learning process perform pretty poorly in this task. This implies that with the help of documents our proposed method is powerful for tag recommendation.

However, the value of precision is a bit low since the groundtruth of testdata does not include all relevant tags. In fact, we crawl the image data from Flickr which is a social image hosting website. Hence the data is not unified and the groundtruth of test data may not be correct. For example, an image with a cat has a tag "people" but has does not have a tag "pet". Consequently, it's necessary for us to do a user study to check the result, in which we randomly select 200 test images and check the top 20 tags. Since this requires much human labor, we are not able to check all 10,000 test images. Six people check the results, and the final precision is the average of these six people. Figure 3(b) displays the results of the user study, which indicates our model can obtain high precision.



(a) Comparison with baselines

(b) The result of user study

Fig. 3. Evaluation of HTLFA

5.4 Impact of Constraint Parameter

In the second experiment, we study the parameter sensitivity of λ_0 on the overall performance of **HTLFA** in tag recommendation. As mentioned before, λ_0 controls the strength of the constraint between P and Q. In this experiment we tune the value of λ_0 to obtain a set of results. Figure 4 shows the recommendation accuracy **P**@10 of **HTLFA** under varying values of λ_0. We find that **HTLFA** performs best and steadily when λ_0 falls at about 0.05 , which implies the document-tag matrix can indeed help learning a more precise latent factor matrix and a soft constraint is better than a hard constraint. Because if λ_0 approaches zero, it means we don't leverage the help of documents, and if λ_0 approaches infinity, it means we are using a hard constraint to force $P \approx Q$ instead of using a soft constraint.

Furthermore, a comparison is made between **HTLFA** and **CMF**(i.e. Collective Matrix Factorization) [21]. In **CMF**, they simultaneously factor several matrices, sharing parameters among factors when an entity participates in multiple relations. That is **CMF** transfers knowledge from a matrix to another by forcing two latent matrices, which are P and Q in our model, to be equal. Actually, equality is a special case of our model. If we let $\lambda_0 = \infty$, P and Q will be forced to be equal, which makes our model more flexible. Since it's not feasible to set $\lambda_0 = \infty$, we directly use two equal latent matrices(i.e. $P = Q$) abandoning the term $\lambda_0\|P-Q\|$, which is called hard constrain. Figure 5 shows that a soft constraint outperforms a hard constraint.

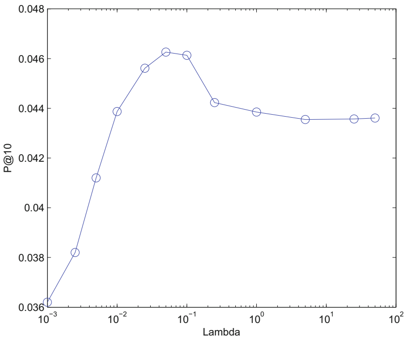


Fig. 4. Varying values of λ_0

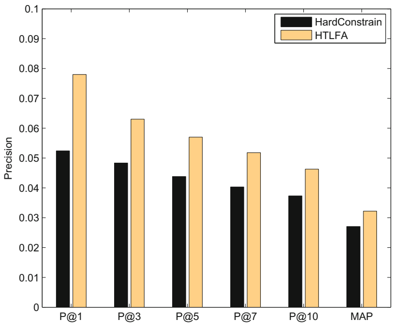


Fig. 5. Comparison between **CMF** and **HTLFA**

5.5 Performance under Longtail Condition

In the third experiment, we also analyze the impact of the amount of training data on the performance of **FA** and **HTLFA** in tag recommendation. We randomly select 30,000, 50,000, 80,000, 120,000 images from all training data to train four models, and evaluate the results. The experimental results are shown in Figure 6. As we can see, compared with the first experiment, the advantage of **HTLFA** becomes larger when training data is sparse. The reason is that when the amount of training data is smaller, the documents can make a larger contribution to supplement the training data. In other words, documents increase the number of heterogeneous training data instances. It properly proves the advantage of transfer learning, that is to remedy the sparsity of the source data. However, **HTLFA** cannot perform well either when training data is extremely sparse. There is no doubt that the distribution of image-tag features is different from the distribution of document-tag features. Hence when the amount of training data approach zero and documents becomes dominant, the model naturally does not work on the task of recommending tags for images.

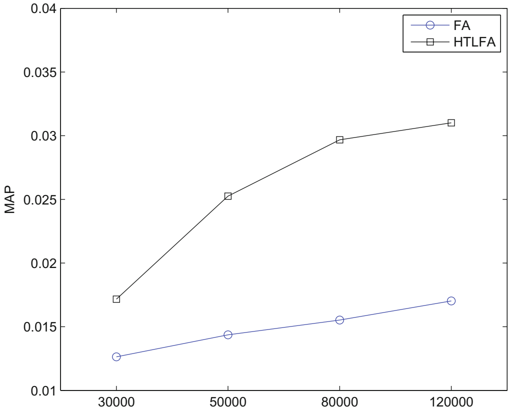
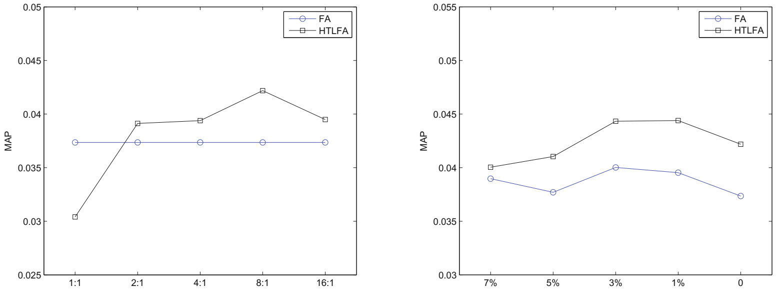


Fig. 6. Varying number of training data

We also compare **HTLFA** and **FA** under the long tail condition. Since words which are in the tail of the frequency distribution for images are almost always in the tail of the distribution for documents, we randomly select 2000 tags and use them to form a long tail tag set. We down sample these tags in images to simulate a long tail condition. We construct several test settings, corresponding to different down sample rates. When evaluating the results, we ignore the groundtruth not included in long tail tag set and focus on recommending long tail groundtruth from the other tags. In Figure 7(a), we vary the iteration number ratio between images and documents fixing the down sample rate at 0. The result shows that $8 : 1$ (i.e., iterate through 8 images before iterating a document) obtains the best performance. Fixing the iteration number ratio at $8 : 1$, Figure 7(b) shows that our model outperforms **FA** under long tail condition. However, as mentioned above, when the down sample rate is extremely low (e.g. zero percent), the shortcoming that the two heterogeneous data sets have different distributions becomes obvious, which finally leads to a decrease in the precision.

5.6 Case Study

Table 3 shows the tag recommendation result of our algorithm **HTLFA** for Flickr data sets. In this figure, six images are from the Flickr data set. Three recommended tags are given at the bottom of each image. From the table we can see that our algorithm can indeed find related tags based on visual contextual information of an image. Here we demonstrate a case that makes a difference among the compared algorithms. Table 4 provides some tags recommended by the four algorithms for a target image about a baby. The top three recommended tags are shown on the right of each algorithm name. As the table shows, **ViCAD** and **NN** have really poor performance on this case. Non-transfer **FA** recommends only one related tag while the top three tags recommended by



(a) Varying the iteration number ratio (b) Varying down sample rate

Fig. 7. Evaluation under long tail condition

Table 3. The **HTLFA** results

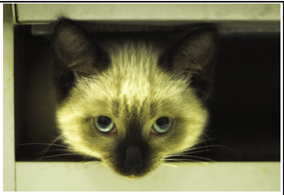
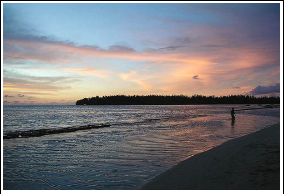
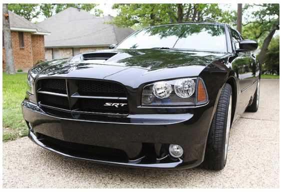
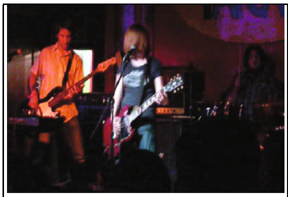
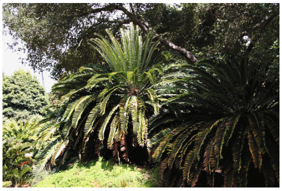
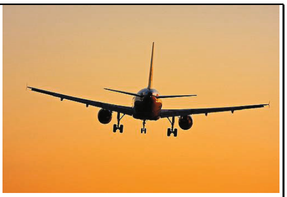
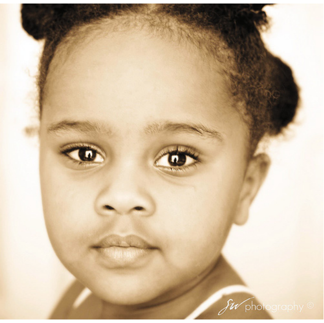
		
cat, grey, pets	clouds, seascape, beach	wheels, voiture, ford
		
concert, music, guitar	nature, wildlife, plant	flight, aviation, plane

Table 4. Tags recommended by the compared algorithms on one case

	**ViCAD**	farriery laminitis arthrosis
	NN	cool surgery rachel
	FA	lips freckles jackson
	HTLFA	lips eyelashes toddler

HTLFA are all related. And we also discover that the tag "toddler" does not appear in the top ten of the recommendation list of **FA**. Hence it is reasonable to believe that it is documents that help discover the related tag "toddler".

6 Conclusion

In this paper, we explore heterogeneous transfer learning for factor alignment integrating documents, tags and images and apply it on the task of tag recommendation by leveraging the help of long text from Wikipedia. We expect to make use of auxiliary text documents to supplement the target domain training data since auxiliary data may reduce the data sparsity in the image domain and word domain. Then we propose a factor alignment ranking model and modify it to make it able to handle the auxiliary text data. Using soft constraints to control the similarity between two latent matrices, we manage to incorporate the information of auxiliary text data in the model. We compare the method with three baselines, and prove that our method outperforms all three. We vary the value of parameter λ_0 to illustrate that a soft constraint is necessary. Comparing our model to one without transfer, we also show that auxiliary text data also improves handling of items from the long tail of the tag frequency distribution. So overall, we have shown that the performance of tag recommendation can be improved by utilizing textual information.

In the future, we will consider other types of auxiliary data as well as more than one data source.

Acknowledgement. We thank Weinan Zhang and Erheng Zhong for discussions. We thank the support of grants from NSFC-RGC joint research project HKUST 624/09 and 60931160445.

References

1. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22, 1345–1359 (2010)
2. Yang, Q., Chen, Y., Xue, G.-R., Dai, W., Yu, Y.: Heterogeneous transfer learning for image clustering via the social web. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: vol. 1, ACL 2009, pp. 1–9. Association for Computational Linguistics, Stroudsburg (2009)
3. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th International Conference on Machine Learning, ICML 2007, pp. 759–766. ACM, New York (2007)
4. Dai, W., Chen, Y., Xue, G.R., Yang, Q., Yu, Y.: Translated learning: Transfer learning across different feature spaces (2008)
5. Zhu, Y., Chen, Y., Lu, Z., Pan, J.S., Xue, G.R., Yu, Y., Yang, Q.: Heterogeneous transfer learning for image classification (2011)
6. Duygulu, P., Barnard, K., de Freitas, J.F.G., Forsyth, D.: Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 97–112. Springer, Heidelberg (2002)

7. Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D.M., Jordan, M.I.: Matching words and pictures. J. Mach. Learn. Res. 3, 1107–1135 (2003)
8. Blei, D.M., Jordan, M.I.: Modeling annotated data. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR 2003, pp. 127–134. ACM, New York (2003)
9. Lavrenko, V., Manmatha, R., Jeon, J.: A model for learning the semantics of pictures (2003)
10. Carneiro, G., Chan, A.B., Moreno, P.J., Vasconcelos, N.: Supervised learning of semantic classes for image annotation and retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(3), 394–410 (2007)
11. Makadia, A., Pavlovic, V., Kumar, S.: A New Baseline for Image Annotation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 316–329. Springer, Heidelberg (2008)
12. Jin, Y., Khan, L., Wang, L., Awad, M.: Image annotations by combining multiple evidence & wordnet. In: Proceedings of the 13th Annual ACM International Conference on Multimedia, MULTIMEDIA 2005, pp. 706–715. ACM, New York (2005)
13. Wang, C., Jing, F., Zhang, L., Zhang, H.-J.: Content-based image annotation refinement. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2007 (2007)
14. Liu, D., Hua, X.S., Wang, M., Zhang, H.J.: Image retagging. In: Proceedings of the International Conference on Multimedia, MM 2010, pp. 491–500. ACM, New York (2010)
15. Liu, D., Yan, S., Hua, X.-S., Zhang, H.-J.: Image retagging using collaborative tag propagation. IEEE Transactions on Multimedia 13(4), 702–712 (2011)
16. Chen, Y., Jin, O., Xue, G.R., Chen, J., Yang, Q.: Visual contextual advertising: Bringing textual advertisements to images (2010)
17. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 2, pp. 524–531 (2005)
18. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2009, Arlington, Virginia, United States. AUAI Press (2009)
19. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60, 91–110 (2004)
20. Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W.: Discovering object categories in image collections (2005)
21. Singh, A.P., Gordon, G.J.: Relational learning via collective matrix factorization. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, pp. 650–658. ACM, New York (2008)

Learning to Perceive Two-Dimensional Displays Using Probabilistic Grammars

Nan Li, William W. Cohen, and Kenneth R. Koedinger

Carnegie Mellon University, Pittsburgh PA 15232, USA,
{nli1,wcohen,koedinger}@cs.cmu.edu

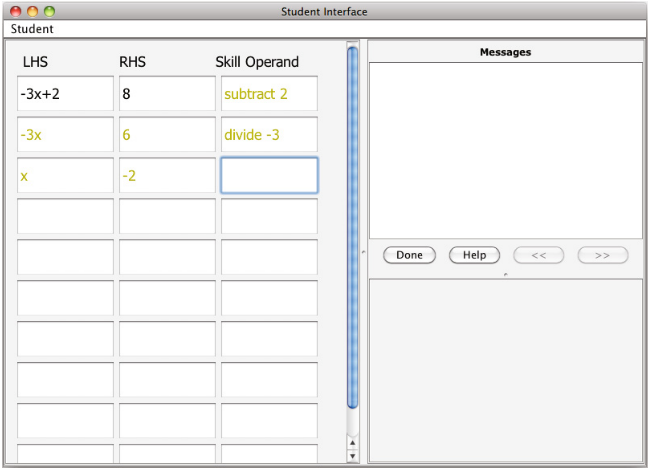
Abstract. People learn to read and understand various displays (e.g., tables on webpages and software user interfaces) every day. How do humans learn to process such displays? Can computers be efficiently taught to understand and use such displays? In this paper, we use statistical learning to model how humans learn to perceive visual displays. We extend an existing probabilistic context-free grammar learner to support learning within a two-dimensional space by incorporating spatial and temporal information. Experimental results in both synthetic domains and real world domains show that the proposed learning algorithm is effective in acquiring user interface layout. Furthermore, we evaluate the effectiveness of the proposed algorithm within an intelligent tutoring agent, *SimStudent*, by integrating the learned display representation into the agent. Experimental results in learning complex problem solving skills in three domains show that the learned display representation is as good as one created by a human expert, in that skill learning using the learned representation is as effective as using a manually created representation.

Keywords: two-dimensional grammar learning, learning to perceive displays, intelligent agent, cognitive modeling.

1 Introduction

Every day, people view and understand many novel two-dimensional (2-D) displays such as tables on webpages and software user interfaces. How do humans learn to process such displays? As an example, Figure 1 shows a screenshot of one interface to an intelligent tutoring system that is used to teach students how to solve algebraic equations. The interface should be viewed as a table of three columns, where the first two columns of each row contain the left-hand side and right-hand side of the equation, and the third column names the skill applied. In tutoring, students enter data row by row, a strategy which requires a correct intuitive understanding of how the interface is organized. SimStudent [1] is a system that uses programming-by-demonstration [2] to develop a rule-based tutor on an arbitrary interface, and to learn effectively, it needs a similar understanding of the way the interface is organized. Incorrect representation of the interface may lead to inappropriate generalization of the acquired skill knowledge, such

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 773–788, 2012.



LHS	RHS	Skill Operand
-3x+2	8	subtract 2
-3x	6	divide -3
x	-2	

Fig. 1. The interface where SimStudent is being tutored in an equation solving domain

as generalizing the skill for adding two numerators to adding two denominators in fraction addition. Past instances of SimStudent have used a hand-coded hierarchical representation of the interface, which is both time-consuming, and less psychologically plausible. Here we consider replacing that hand-coded element with a learned representation.

More generally, we consider using a two-dimensional variant of a probabilistic context-free grammar (pCFG) to model how a user perceives the structure of a user interface, and propose a novel 2-D pCFG learning algorithm to model acquisition of this representation. Our learning method exploits both the spatial layout of the interface, and temporal information about when users interact with the interface. The alphabet of the grammar is a vocabulary of symbols representing primitive interface-element types. For example, in Figure 1, the type of the cells in the first two columns is *Expression*, and the type of the last cell in the each column is *Skill*. (In SimStudent, these primitive types can be learned from prior experience.) We extend an ordinary one-dimensional (1-D) pCFG learner [3] to acquire two-dimensional grammar rules, using a two-dimensional probabilistic version of the Viterbi training algorithm to learn parameter weights and a structure hypothesizer that uses spatial and temporal information to propose grammar rules.

We then integrate this two-dimensional representation learner into SimStudent. SimStudent is used to model the learning of human students in tutoring domains such as algebra. Many students learn quickly, from few examples; however, some learn more slowly. Previous work in cognitive science [4] showed that one of the key factors that differentiates experts and novices in a field is their different prior knowledge of world state representation. Previously, we had to manually encode such representation, which is both time consuming and error prone. We now extend SimStudent by replacing the hand-coded display representation with the statistically learned display representation. We demonstrate the proposed algorithm in tutoring systems, and for simplicity will refer to terminal symbols in the grammar as interface element, but we emphasize that the proposed algorithm should work for two-dimensional displays of other types as well. We evaluate the proposed algorithms in both synthetic domains and real world domains, with and without integration into SimStudent. Experimental results show that the proposed learning algorithm is effective in acquiring user interface layouts. The SimStudent with the proposed representation learner acquired domain knowledge at similar rates to a system with hand-coded knowledge. The main contribution of this paper is to use probabilistic grammar induction to model learning to perceive two-dimensional visual displays.

2 Related Work

In previous work, we have developed a one-dimensional (1-D) pCFG learner to acquire representations of 1-D strings (e.g., the parse structure of *-3x*), and showed that the acquired representations yield effective learning, while reducing the amount of knowledge engineering required in building an intelligent agent [5]. Moreover, it has been shown that with this extension, the intelligent agent becomes a better model of human students [6], and can be used to better understand human student learning behavior [7]. In this work, we further extend the representation learner to acquire representations in a 2-D space using a two-dimensional variant of pCFG.

One closely related research area that also uses two-dimensional pCFGs is learning to recognize equations (e.g., [8, 9]). Algorithms in this direction often assume the structure of the grammar is given, and use a two-dimensional parsing algorithm to find the most likely parse of the observed image. Our system differs from their approaches in that we model the acquisition of the grammar structure, and apply the technique to another domain, learning to perceive user interface.

Research on extracting structured data on the web (e.g., [10–12]) shares a clear resemblance with our work, as it also concerns on understanding structures embedded in a two-dimensional space. It differs from our work in that webpages have an observable hierarchical structure in the form of their HTML parse trees, whereas we only observe the 2-D visual displays, which have no such structural information.

3 Problem Definition

To learn the representation of a 2-D display, we first need to formally define the input and output of the problem.

3.1 Input

The input to the algorithm is a set of records, $\mathcal{R} = \{R_1, R_2, ..., R_n\}$, associated with examples shown on the display observed by people. Figure 1 shows one problem example in this algebra tutor interface. Each record, R_i $(i = 1, 2, ...n)$, records how and when the elements in the display are filled out by users. Thus, R_i is a sequence of tuples, $\langle T_{i1}, T_{i2}, ..., T_{im}\rangle$, where each tuple, T_{ik} $(k = 1, 2, ..., m)$, is associated with one display element that is used in solving the problem. The tuples in a record are ordered by time. For example, to solve the problem, *-3x+2 = 8*, shown in Figure 1, the cells in the first three rows (except for the last cell of the third row) are used. We do not assume that meta-elements such as columns and rows are given, but we will assume that each display element occupies a rectangular region, and that we can detect when regions are adjacent. In this case, R_i will contain 12 tuples, $\langle T_{i1}, T_{i2}, ..., T_{i12}\rangle$, that correspond to the eight cells, *Cell 11*, *Cell 12*, *Cell 13*, *Cell 21*, *Cell 22*, *Cell 23*, *Cell 31*, and *Cell 32*, and the four buttons, *done*, *help*, <<, and >>.

Each tuple consists of seven items,

$$T_{ik} = \langle type, x_{left}, x_{right}, y_{up}, y_{bottom}, timestamp_{start}, timestamp_{end}\rangle$$

where *type* is the type of the input to the display element, x_{left}, x_{right}, y_{up}, and y_{bottom} define the x and y coordinates of the space the element ranges over, and $timestamp_{start}$ and $timestamp_{end}$ are the start and ending time when the display element is filled out by the user. For example, given the problem *-3x+2 = 8*, the tuple associated with *Cell 11* is $T_{i1} = \langle Expression, 0, 1, 0, 1, 0, 0\rangle$. The timestamp of *Cell 11* is *0*, since both *Cell 11* and *Cell 21* were entered first by the tutor as the given problem. As mentioned above, we have developed a 1-D pCFG learner that acquires parse structures of 1-D strings. The type of the input is the non-terminal symbol associated with the parse tree of the content. Hence, the type of *-3x+2* is *Expression.*

3.2 Output

Given the input, the objective of the grammar learner is to acquire a 2-D pCFG, $\mathcal{G}$, that best captures the structural layout given the training records, that is,

$$\arg\max_{\mathcal{G}} \; p\left(\mathcal{R} \mid \mathcal{G}\right)$$

under the constraint that all records share the same parse structure (i.e., layout). We will explain this in more detail in the algorithm description section.

The output of the layout learner is a two-dimensional variant of pCFG [8], which we define below. When used to parse a display, this grammar will generate a tree-like hierarchical grouping of the display elements.

Two-Dimensional pCFG. 2-D pCFG is an extended version of 1-D pCFG. Each 2-D pCFG, $\mathcal{G}$, is defined by a four-tuple, $\langle \mathcal{V}, \mathcal{E}, \mathcal{R}ules, S\rangle$. $\mathcal{V}$ is a finite set of non-terminal symbols that can be further decomposed to other non-terminal or terminal symbols. $\mathcal{E}$ is a finite set of terminal symbols, that makes up the actual content of the "2-D sentence". In our algebra example, the terminal symbols of the visual display are the input types associated with the display elements

Table 1. Part of the two-dimensional probabilistic context free grammar for the equation solving interface

Terminal symbols: *Expression*, *Skill*;
Non-terminal symbols: **Table**, **Row**, **Equation**, **Exp**, **Ski**
Table → 0.7, [*v*] **Table Row**
Table → 0.3, [*d*] **Row**
Row → 1.0, [*h*] **Equation Ski**
Equation → 1.0, [*h*] **Exp Exp**
Exp → 1.0, [*d*] *Expression*
Ski → 0.5, [*d*] *Skill*

(e.g., *Expression*, *Skill*). $\mathcal{R}ules$ is a finite set of 2-D grammar rules. S is the start symbol.

Each 2-D grammar rule is of the form

$$V \rightarrow p, [direction]\ \gamma_1\ \gamma_2\ ...\gamma_n$$

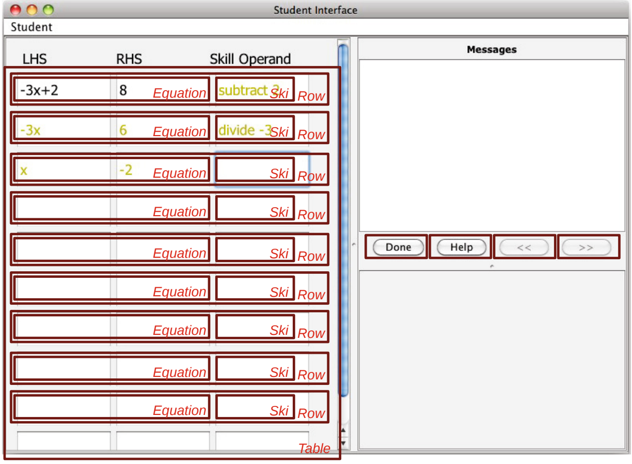
where $V \in \mathcal{V}$, p is the probability of the grammar rule used in derivations[1], and $\gamma_1, \gamma_2, ...\gamma_n$ is either a sequence of terminal symbols or a sequence of non-terminal symbols. Without loss of generality, in this case, we only consider grammar rules that have one or two symbols at the right side of the arrow.

direction is a new field added for the 2-D grammar. It specifies the spatial relation among its children. The value of the direction field can be d, h, or v. d is the default value set for grammar rules that have only one child, in which case there is no direction among the children. h (v) means the children generated by the grammar rule should be placed horizontally (vertically) with respect to each other. An example of a two-dimensional pCFG of the equation solving interface is shown in Table 1[2]. The corresponding layout is presented in Figure 2. The rows in the table are placed vertically with respect to other rows. Thus, the direction field in the grammar rule "**Table** → 0.7, [*v*] **Table Row**" is set to be v. On the other hand, the equation should be placed horizontally with the skill cell in the third column, so the direction field of "**Row** → 1.0, [*h*] **Equation Ski**" is h. These three direction values form the original direction value set.

Since the interface elements may not form a rectangle sometimes (e.g., the table and the buttons in the equation solving interface), we further extend the direction field to have two additional values pv and "ph". pv (ph) means that the children of the grammar rule should be placed vertically (horizontally) with respect to each other, but the parts in the interface associated with these children do not have to form a rectangle. As shown in Figure 2, the table in the left side and the buttons in the right side can be placed horizontally, but do not form

[1] The sum of the probabilities associated with rules that share the same head, V, equals to 1.

[2] The non-terminal symbols are replaced with meaningful names here. The symbols in the learned grammars are synthetic-generated symbols.



LHS	RHS	Skill Operand
-3x+2	8	subtract 2
-3x	6	divide -3
x	-2	

Fig. 2. An example layout of the interface where SimStudent is being tutored in an equation solving domain

a rectangle. In this case, the grammar rule should use *ph* instead of *h* as the directional field value. These direction values are less-preferred than the original values. Grammar rules that have such direction values will only be added if no more rules with directions *d*, *h*, or *v* can be found.

Layout. Given the 2-D pCFG, the final output of the display representation is a hierarchical grouping of the display elements, which we will call a layout, L. Figure 2 shows an example layout of the equation solving interface. The left side of the interface contains a row-ordered table, where each row is further divided into an equation and a skill. The right side of the interface contains a list of buttons that can be pressed by students to ask for help or to indicate when he/she considers the problem is solved.

4 Learning Two-Dimensional Display Layout Using Probabilistic Grammars

Now that we have formally defined the learning task, we are ready to describe the 2-D display layout learner. Recently, we have proposed a 1-D grammar learner [3], and have shown that the 1-D grammar learner acquires knowledge more effectively and runs faster than the inside-outside algorithm [13][3]. Hence, we further extend the one-dimensional grammar learner to acquire a 2-D pCFG from two-dimensional training records.

Algorithm 1 shows the pseudo code of the 2-D display layout learner. The learning algorithm iterates between a greedy structure hypothesizer (GSH) and a Viterbi training phase. The GSH tries to construct non-terminal symbols as well

[3] rakaposhi.eas.asu.edu/nan-tist.pdf.

Algorithm 1. *2D-Layout-Learner* constructs a set of grammar rules, $\mathcal{R}ules$, from the training records, $\mathcal{R}$, and a set of terminal symbols $\mathcal{E}$.

```
Input: Record Set $\mathcal{R}$, Terminal Symbol Set $\mathcal{E}$
$\mathcal{R}ules := \phi$;
while not-all-records-have-one-layout($\mathcal{R}$, $\mathcal{R}ules$) do
    $\mathcal{R}ules :=$ GSH($\mathcal{R}$, $\mathcal{E}$, $\mathcal{R}ules$);
    $\mathcal{R}ules :=$ Viterbi-training($\mathcal{R}$, $\mathcal{R}ules$);
end
return $\mathcal{R}ules$
```

as grammar rules that could parse all input records, $\mathcal{R}$. The set of constructed rules are then set as the start point for the Viterbi training algorithm. Next, the Viterbi training algorithm iteratively re-estimates the probabilities associated with all grammar rules until convergence. If the grammar rules are not sufficient in generating a layout in the Viterbi training algorithm, GSH is called again to add more grammar rules. This process continues until at least one layout can be found.

Since an appropriate way of transferring previously acquired knowledge to later learning process could potentially improve the learning speed, we further designed a learning mechanism that transfers the acquired grammar with the application frequency of each rule from previous tasks to future tasks. Due to the limited space, we will not present the detail of this extension in this paper.

4.1 Viterbi Training

Given a set of grammar rules from the GSH step, the Viterbi training algorithm tunes the probabilities on the grammar set, and removes unused rules.[4] We consider an iterative process. Each iteration involves two steps.

One key difference between learning the parse trees of 1-D strings and learning the GUI element layout is that the parse trees for different input contents are different (e.g., *-3x* vs. *5x+6*), whereas the GUI elements should always be organized in the same way even if the input contents in the GUI elements have changed from problem to problem. For instance, students will always perceive the equation solving interface as multiple rows, where each row consists of an equation along with a skill operation, no matter which problem they are given. Therefore, instead of finding a grammar that parses the interface given specific input, the learning algorithm should acquire one layout for the interface across different problems. This effectively adds a constraint on the learning algorithm.

In the first step, the algorithm computes the most probable parse trees, $\mathcal{T}$, for all training records using the current rules, under the constraint that the parse structure among these trees should be the same, that is,

[4] More detailed discussion on why a Viterbi training algorithm instead of the standard CKY is used can be found in [14], which is mainly because of overfitting.

$$\begin{aligned}\mathcal{T} &= \arg\max_{\mathcal{T}}\ p\left(\mathcal{T} \mid \mathcal{R}, \mathcal{G}, S\right)\\ &= \bigcup_{i=1,2,\dots n} \arg\max_{T_i}\ p\left(T_i \mid R_i, \mathcal{G}, S\right)\\ &s.t.\ \ parse(T_1) = parse(T_2) = ... = parse(T_n)\ \forall\ T_i \in \mathcal{T}\end{aligned}$$

where T_i is the parse tree with root S for record R_i given the current grammar $\mathcal{G}$, and $parse(T_i)$ denotes the parse structure of T_i ignoring the symbols associated with the parse nodes[5]

Since any subtree of a most probable parse tree is also a most probable parse subtree, we have

$$\begin{aligned}&p\left(T_i \mid R_i, \mathcal{G}, S_i\right)\\ &= \max_{rule, idx} \begin{cases} p\left(rule \mid \mathcal{G}\right) \times p\left(T_{i,1} \mid R_{i,1}, \mathcal{G}, S_{i,1}\right) \times p\left(T_{i,2} \mid R_{i,2}, \mathcal{G}, S_{i,2}\right) \\ \qquad \text{if } rule \text{ is } S_i \rightarrow p(rule|\mathcal{G}), [direction]\ S_{i,1}\ S_{i,2}, \\ p\left(rule \mid \mathcal{G}\right) \times p\left(T_{i,1} \mid R_i, \mathcal{G}, S_{i,1}\right) \\ \qquad \text{if } rule \text{ is } S_i \rightarrow p(rule|\mathcal{G}), [direction]\ S_{i,1}, \\ p\left(rule \mid \mathcal{G}\right) \\ \qquad \text{if } rule \text{ is } S_i \rightarrow p(rule|\mathcal{G}), [direction]\ E_{i,1}, \text{ and } E_{i,1} \in \mathcal{E}. \end{cases}\end{aligned}$$

where $rule$ is the rule that is used to parse the current record R_i, $p\left(rule \mid \mathcal{G}\right)$ is the probability of $rule$ used among all grammar rules (in all directions) that have head S_i, $R_{i,1}$ and $R_{i,2}$ are the split traces based on the direction of the rule, *direction*, and the place of the split, *idx*, and T_i, $T_{i,1}$ and $T_{i,2}$ are the most probable parse trees for R_i, $R_{i,1}$ and $R_{i,2}$ respectively. Using this recursive equation, the algorithm builds the most probable parse trees in a bottom-up fashion.

After getting the parse trees for all records, the algorithm moves on to the second step. In this step, the algorithm updates the selection probabilities associated with the rules. For a rule with head V, the new probability of getting chosen is simply the total number of times that rule appearing in the Viterbi parse trees divided by the total number of times that V appears in the parse trees, that is,

$$p(rule_i|\mathcal{G}) = \frac{|rule_i \text{ appearing in parse trees}|}{|V_i \text{ appearing in parse trees}|}$$

where $rule_i$ is of the form $V_i \rightarrow p, [direction], \gamma_1, \gamma_2, ...\gamma_n$, $n = 1$ *or* 2.

After finishing the second step, the algorithm starts a new iteration until convergence. This learning procedure is a fast approximation of expectation-maximization, which approximates the posterior distribution of trees given parameters by the single MAP hypothesis. The output of the algorithm is an

[5] In the case that some record uses less elements than the other records (e.g., simpler problems that require less steps), $parse(T_i)$ is considered equal to $parse(T_j)$ as long as the parse structures of the shared elements are the same.

Algorithm 2. *GSH* constructs a set of grammar rules, $\mathcal{R}ules$, a set of terminal symbols $\mathcal{E}$, and from the training records, $\mathcal{R}$.

```
Input: Record Set $\mathcal{R}$, Terminal Symbol Set $\mathcal{E}$, Grammar Rule Set $\mathcal{R}ules$
if is-empty-set($\mathcal{R}ules$) then
    $\mathcal{R}ules$ := generate-terminal-grammar-rules($\mathcal{E}$);
end
while not-all-records-are-parsable($\mathcal{R}$, $\mathcal{R}ules$) do
    if has-recursive-structure($\mathcal{R}$) then
        rule := generate-recursive-rule($\mathcal{R}$);
    else
        rule := generate-most-frequent-non-added-rule($\mathcal{R}$);
    end
    $\mathcal{R}ules$ := $\mathcal{R}ules$ + rule;
    $\mathcal{R}$ := update-record-set-with-rule($\mathcal{R}$, rule, $\mathcal{R}ules$);   // First, update the
    record set using rule; second, update the record set using all
    acquired $\mathcal{R}ules$
end
$\mathcal{R}ules$ = initialize-probabilities($\mathcal{R}ules$);
return $\mathcal{R}ules$
```

updated 2-D pCFG, $\mathcal{G}$, and the most probable layout of the interface. For elements that have never been used in the training examples, the acquired layout will not include them in it as there is no information for them in the record. But the acquired grammar may be able to generalize to those elements. For example, if the acquired grammar learns a recursive rule across rows, it will be able to generalize to more rows than the training records have reached.

The complexity of the Viterbi training phase is $\mathcal{O}(|iter| \times |\mathcal{R}| \times |\mathcal{R}ules_{nt}| \times |\max R_i.length|!)$, where $|iter|$ is the number of iterations, $|\mathcal{R}|$ is the number of records, $|\mathcal{R}ules_{nt}|$ is the number of rules that reduce to non-terminal symbols, $|\max R_i.length|$ is the length of the longest record. In practice, since the number of rules generated by GSH is small, and we cache previously calculated parse trees in memory, as we will see in the experiment section, all learning tasks are completed within a reasonable amount of time.

4.2 Greedy Structure Hypothesizer (GSH)

As with the standard Viterbi training algorithm, the output of the algorithm converges toward only a local optimum. It often requires more iterations to converge if the starting point is not good. Moreover, since the complexity of the Viterbi training phase increases as the number of grammar rules increases, we designed a greedy structure hypothesizer (GSH) that greedily adds grammar rules for frequently observed "adjacent" symbol pairs. Note that instead of building a structure learner from scratch, we extend an existing one [3] to accommodate the 2-D space. Extending other learning mechanisms is also possible. To formally define adjacency, let's first define two terms, *temporally adjacent*, and *horizontally (vertically) adjacent.*

Definition 1. *Two tuples, T_{i1} and T_{i2}, are temporally adjacent, iff the two tuples' time intervals overlap, i.e.*

$$[T_{i1}.timestamp_{start}, T_{i1}.timestamp_{start}) \cap$$
$$[T_{i2}.timestamp_{start}, T_{i2}.timestamp_{start}) \neq \emptyset$$

Definition 2. *Two tuples, T_{i1} and T_{i2}, are horizontally adjacent, iff the spaces taken up by the two tuples are horizontally next to each other, and form a rectangle, i.e.*

$$T_{i1}.x_{right} = T_{i2}.x_{left} \text{ or } T_{i2}.x_{right} = T_{i1}.x_{left}$$
$$T_{i1}.y_{up} = T_{i2}.y_{up}$$
$$T_{i1}.y_{bottom} = T_{i2}.y_{bottom}$$

Definition 3. *Two tuples, T_{i1} and T_{i2}, are vertically adjacent, iff the spaces took up by the two tuples are vertically next to each other, and form a rectangle, i.e.*

$$T_{i1}.y_{bottom} = T_{i2}.y_{up} \text{ or } T_{i2}.y_{bottom} = T_{i1}.y_{up}$$
$$T_{i1}.x_{left} = T_{i2}.x_{right}$$
$$T_{i1}.x_{right} = T_{i2}.x_{left}$$

Now, we can define what is a *2D-mergeable pair.*

Definition 4. *Two tuples, T_{i1} and T_{i2}, are 2D-mergeable, iff the two tuples are both temporally adjacent and horizontally (vertically) adjacent.*

The structure hypothesizer learns grammar rules in a bottom-up fashion. The pseudo code of the structure hypothesizer is shown in Algorithm 2. The grammar rule set, $\mathcal{R}ules$, is initialized to contain rules associated with terminal symbols, when GSH is called for the first time. Then the algorithm detects whether there are recursive structures embedded in the records (e.g., **Row**, **Row**, ...**Row**) , and learns a recursive rule for it if finds one (e.g., **Table** → 0.7, $[v]$ **Table Row**). If the algorithm fails to find recursive structures, it starts to search for the 2D-mergeable pair (e.g., ⟨**Equation**, **Ski**⟩) that appears in the record set most frequently, and constructs a grammar rule (e.g., **Row** → 1.0, $[h]$ **Equation Ski**) for that 2D-mergeable pair. The direction field value is set based on whether the 2D-mergeable pairs are horizontally or vertically adjacent. If the Viterbi training phase cannot find a layout based on these rules, less frequent pairs are added later. When there is no more pair that is 2D-mergeable, it is possible that some training record has not been fully parsed, since some symbol pairs that are horizontally (vertically) ordered may not form rectangles. The grammar rules constructed for these symbol pairs in this case will use the extended direction values (e.g., ph, pv). After getting the new rule, the system updates the current record set with this rule by replacing the pairs in the records with the head of the rule.

After learning the grammar rules, the GSH assigns probabilities associated with these grammar rules. For each rule with head V, p is assigned to 1 divided by the number of rule that have V as the head. In order to break the symmetry among all rules, the algorithm adds a small random number to each probability and normalizes the values again. This structure learning algorithm provides a redundant set of grammar rules to the Viterbi algorithm.

5 Experimental Results of the Two-Dimensional Display Learner

In order to evaluate whether the proposed layout learner is able to acquire the correct layout, we carried out three experiments in progressively more realistic settings. All experiments were performed on a machine with a 3.06 GHz CPU and 4 GB Memory. The time the layout learner takes to learn ranges from less than 1 millisecond to 442 milliseconds per training record.

5.1 Experiment Design

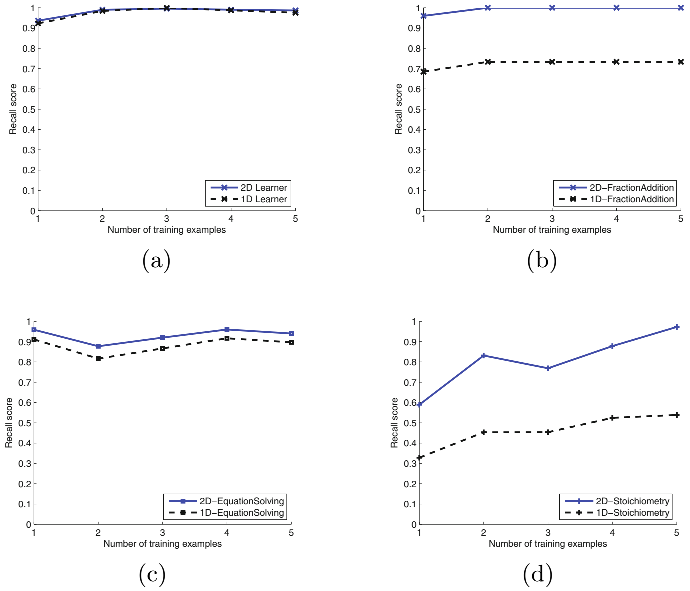
In this section, we use the 1-D layout learner (i.e., 1-D pCFG learner) as a baseline, and compare it with the proposed 2-D layout learner. In order to make the training records learnable by the 1-D layout learner, we first transform each training record into a row-ordered 1-D record, and then call the 1-D layout learner on the transformed records.

We evaluate the quality of the learned parses with the most widely-used evaluation measurements [15]: (1) the *Crossing Parentheses* score, which is the number of times that the learned parse has a structure such as ((A B) C) and the oracle parse has one or more structures such as (A (B C)) which "cross" with the learned parse structure; (2) the *Recall* score, which is the number of parenthesis pairs in the intersection of the learned and oracle parses (L intersection O) divided by the number of parenthesis pairs in the oracle parse O, i.e., (L intersection O) / O. To better understand the crossing parentheses score, we further normalize it so that it ranges from zero to one.

5.2 Experiments in Randomly Generated Synthetic Domains

In the first experiment, we randomly generate 50 oracle two-dimensional grammars. For each oracle grammar, we randomly generate a sequence of 15 training layouts[6] based on the oracle grammar. Each randomly-generated oracle grammar forms an and-or tree, where each non-terminal symbol can be decomposed by either a non-recursive or a recursive rule. Each grammar has 50 non-terminal symbols in it. For each layout, we give the layout learners a fixed number of training records. The two layout learners (i.e., the 1-D layout learner with row-based transformation and the 2-D layout learner) are trained on the 15 layouts sequentially using a transfer learning mechanism developed for the layout learner. The

[6] Some layouts may be the same.



(a) (b)

(c) (d)

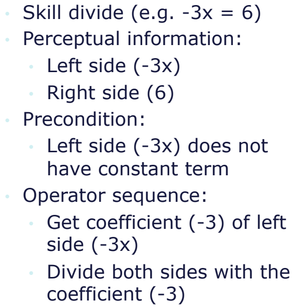
Fig. 3. Recall scores in a) randomly-generated domains, and three synthetic domains, b) fraction addition, c) equation solving, d) stoichiometry

transfer learning mechanism is not described here due to the limited space. Then, we generate another layout with a fixed number of testing records by the oracle grammar, and test whether the grammars acquired by the two layout learners are able to correctly parse the testing records.

Figure 3(a) presents the recall scores of the layout learners averaged over 50 grammars. Both learners perform surprisingly well. They are able to achieve close to one recall scores, and close to zero crossing parentheses scores with only five training examples per layout. To better understand the result, we take a close look at the data. Since the oracle grammar is randomly generated, the probability of getting a hard-to-learn grammar is very low. In fact, many of the training records are traces of single rows or columns, which makes learning easy. Hence, to challenge the layout learner more, we carried out a second experiment.

5.3 Experiments in Three Synthetic Domains

We examine three tutoring systems used by human students: fraction addition, equation solving, and stoichiometry, and manually construct an oracle grammar that is able to parse these three domains. Moreover, the oracle grammar can further generate variants of the existing user interfaces. For example, instead of adding two fractions together, the oracle grammar can generate interfaces that can be used to add three factions. We carry out the same training process based on this manually-constructed oracle grammar, and test the quality of the acquired grammar in three domain variants.



- Skill divide (e.g. -3x = 6)
- Perceptual information:
  - Left side (-3x)
  - Right side (6)
- Precondition:
  - Left side (-3x) does not have constant term
- Operator sequence:
  - Get coefficient (-3) of left side (-3x)
  - Divide both sides with the coefficient (-3)

Fig. 4. Original and extended production rules for divide in a readable format

The interface of the fraction addition tutor has four rows, where the upper two rows are filled with the problem (e.g., $\frac{3}{5} + \frac{2}{3}$), and the lower two rows are empty cells for the human students to fill in. The equation solving tutor's interface is shown in Figure 1. The interface of the stoichiometry domain contains four tables of different sizes. The four tables are used to provide given values, to perform conversion, to self-explain for the current step, and to compute intermediate results. All tables are of column-based orders.

Figure 3(b), 3(c), 3(d) show the recall scores of the three domains averaged over 50 runs. Both learners achieve better performance with more training examples. We also see that the 2-D layout learner has significantly ($p < 0.0001$) higher recall scores than the 1-D layout learner in all three domains. Both fraction addition and stoichiometry contain tables/subtables of column-based orders. The row-based transformation of the 1-D layout learner removes the column information, and thus hurts the learning performance. The crossing parentheses scores for both learners are always close to zero across three domains, which indicates the acquired grammar does not generate bad "crosses" often.

6 Experimental Results within an Intelligent Agent

In order to understand how display representation learning affects agent learning effectiveness, the last experiment that we carry out is within an intelligent agent, *SimStudent*. SimStudent is an intelligent agent that inductively learns skills to solve problems from demonstrated solutions and from problem solving experience. It is an extension of programming by demonstration [2] using inductive logic programming [16] as an underlying learning technique.

Given a sequence of problem examples, the knowledge acquired by SimStudent defines "where" to look for useful information in the GUI, and "when" the useful information satisfies certain conditions, "how" to proceed. This skill knowledge is represented as production rules. Figure 4 shows an example of a production rule learned by SimStudent in its readable format[7]. The perceptual information

[7] Actual production rules follows the LISP format.

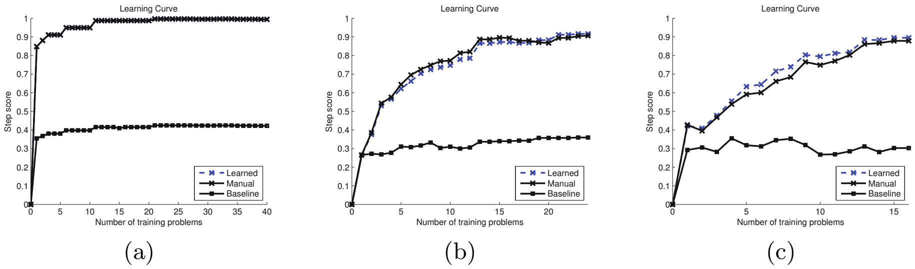


Fig. 5. Learning curves of three SimStudents in three domains, a) fraction addition, b) equation solving, c) stoichiometry

part is acquired by the "where" learner. The precondition part is learned by the "when" learner. The operator function sequence part is created by the "how" learner. The rule to "divide both sides of $-3x = 6$ by -3" shown in Figure 4 would be read as "given a left-hand side (i.e., $-3x$) and a right-hand side (6) of the equation, when the left-hand side does not have a constant term, then get the coefficient of the term of the left-hand side and divide both sides by the coefficient." The "where" learner requires the layout of the interface to be given as input, which is essential for constraining the search space of the other two learning components. Previously, the agent developers need to manually encode such layout as prior knowledge, which hurts the usability of SimStudent as an authoring tool for building cognitive tutors, and fails to model display representation learning. With the layout learner, we are now able to acquire the layout based on the training problems SimStudent observes.

6.1 Experiment Design

We use the actual tutor interfaces in three tutoring domains. The 2-D layout learner is first trained on no more than five problems used to tutor human students, and sends its output to SimStudent. An automatic tutor (also used by human students) then teaches the SimStudent with the constructed/acquired layouts with one set of problems, and tests SimStudents' performance on another set of problems. Both the training and testing problems are problems used by human students. In each domain, SimStudent is trained on 12 problem sequences. Three SimStudents are compared in the experiment. One SimStudent (*manual*) is given the manually-constructed layout, one SimStudent (*learned*) is given the acquired layout, and one SimStudent (*baseline*) is given a row-based layout[8].

To measure learning gain, we calculated a *step score* for each step in the testing problem. Among all possible correct next steps, we counted the number of correct steps that were actually proposed by some applicable production rule, and reported the step score as the number of the correct next steps proposed by learned rules divided by the total number of correct next steps plus the number

[8] A fully flat layout performs so badly that SimStudent cannot finish learning.

of incorrect next steps proposed. For example, if there were four possible correct next steps, and SimStudent proposed three, of which two were correct, and one was incorrect, then only two correct next steps were covered, and thus the step score is *2/(4+1)=0.4*. Step score measures both recall and precision of the proposed next steps. We report the average step score over all testing problem steps for each curriculum.

6.2 Results

Figure 5 shows the learning curves of the three SimStudents across three domains. In all three cases, the SimStudent with a row-based layout (*baseline*) performs significantly ($p < 0.0001$) worse than the other two SimStudents. This shows the importance of the layout in achieving effective learning. Both the SimStudent with the manually-constructed layout (*manual*) and the SimStudent with the learned layout (*learn*) perform well across three domains. There is no significant difference between the two SimStudents, which suggests that the acquired layouts are as good as the manually constructed layouts.

7 Future Work

Although in this paper, we mainly focus on using the two-dimensional grammar learner to model interface layouts, the algorithm is not limited to this specific task. We would like to explore the generality of the proposed approach in other tasks. Reading tables on webpages or notes on paper are potentially interesting tasks. Sometimes, notes on a paper may not be well-aligned. In this case, the layout learning algorithm will need to be able to align these contents.

Moreover, we would like to test whether the layout learner can be used to recognize two-dimensional complex math equations. Correct 2-D layouts of tables are also important in completing calculation tasks in Excel. We would like to see whether the 2-D grammar learner can be used to help learning to perform tasks in Excel.

Finally, the complexity of the current Viterbi training algorithm increases rapidly with the lengths of the training records. Although the GSH and the caching mechanism speed up the learning process a lot, we would like to further optimize the Viterbi training phase to ensure scalability of the learning algorithm.

8 Concluding Remarks

In summary, we proposed a novel approach that models learning to perceive visual displays by grammar induction. More specifically, we extend an existing one-dimensional pCFG learning algorithm to support acquisition of a two-dimensional variant of pCFG by incorporating spatial and temporal information. We showed that the two-dimensional layout learner is more effective than the one-dimensional layout learner in general. When integrated into an intelligent agent, the SimStudent using the acquired layouts performs equally well comparing with the SimStudent given manually constructed layouts.

References

1. Li, N., Matsuda, N., Cohen, W.W., Koedinger, K.R.: Integrating representation learning and skill learning in a human-like intelligent agent. Technical Report CMU-MLD-12-1001, Carnegie Mellon University (January 2012)
2. Lau, T., Weld, D.S.: Programming by demonstration: An inductive learning formulation. In: Proceedings of the 1999 International Conference on Intelligence User Interfaces, pp. 145–152 (1998)
3. Li, N., Cohen, W.W., Koedinger, K.R.: A computational model of accelerated future learning through feature recognition. In: Proceedings of 10th International Conference on Intelligent Tutoring Systems, pp. 368–370 (2010)
4. Chi, M.T.H., Feltovich, P.J., Glaser, R.: Categorization and representation of physics problems by experts and novices. Cognitive Science 5(2), 121–152 (1981)
5. Li, N., Cohen, W.W., Koedinger, K.R.: Efficient cross-domain learning of complex skills. In: Proceedings of the 11th International Conference on Intelligent Tutoring Systems (2012)
6. Li, N., Matsuda, N., Cohen, W.W., Koedinger, K.R.: A machine learning approach for automatic student model discovery. In: Proceedings of the 4th International Conference on Educational Data Minin., pp. 31–40 (2011)
7. Li, N., Cohen, W.W., Koedinger, K.R.: Problem order implications for learning transfer. In: Proceedings of the 11th International Conference on Intelligent Tutoring Systems (2012)
8. Chou, P.A.: Recognition of Equations Using a Two-Dimensional Stochastic Context-Free Grammar. In: Proceedings of Visual Communications and Image Processing, vol. 1199, pp. 852–863 (November 1989)
9. Vanlehn, K.: Learning one subprocedure per lesson. Artificial Intelligence 31, 1–40 (1987)
10. Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 337–348. ACM, New York (2003)
11. Cafarella, M.J., Halevy, A.Y., Wang, D.Z., Eugene, W., Zhang, Y.: Webtables: exploring the power of tables on the web. Proceedings of the VLDB Endowment 1(1), 538–549 (2008)
12. Crescenzi, V., Mecca, G., Merialdo, P.: Roadrunner: Towards automatic data extraction from large web sites. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 109–118. Morgan Kaufmann Publishers Inc., San Francisco (2001)
13. Lari, K., Young, S.J.: The estimation of stochastic context-free grammars using the inside-outside algorithm. Computer Speech and Language 4, 35–56 (1990)
14. Li, N., Cushing, W., Kambhampati, S., Yoon, S.: Learning probabilistic hierarchical task networks as probabilistic context-free grammars to capture user preferences. Technical Report arxiv:1006.0274 (Revised), Arizona State University (2011)
15. Harrison, P., Abney, S., Black, E., Gdaniec, C., Grishman, R., Hindle, D., Ingria, R., Marcus, M.P., Santorini, B., Strzalkowski, T.: Evaluating syntax performance of parser/grammars of English. In: Natural Language Processing Systems Evaluation Workshop. Technical Report, Griffis Air Force Base, NY, pp. 71–78 (1991)
16. Muggleton, S., de Raedt, L.: Inductive logic programming: Theory and methods. Journal of Logic Programming 19, 629–679 (1994)

Transfer Spectral Clustering

Wenhao Jiang and Fu-lai Chung

Department of Computing, Hong Kong Polytechnic University,
Hunghom, Kowloon, Hong Kong
{cswhjiang,cskchung}@comp.polyu.edu.hk

Abstract. Transferring knowledge from auxiliary datasets has been proved useful in machine learning tasks. Its adoption in clustering however is still limited. Despite of its superior performance, spectral clustering has not yet been incorporated with knowledge transfer or transfer learning. In this paper, we make such an attempt and propose a new algorithm called transfer spectral clustering (TSC). It involves not only the data manifold information of the clustering task but also the feature manifold information shared between related clustering tasks. Furthermore, it makes use of co-clustering to achieve and control the knowledge transfer among tasks. As demonstrated by the experimental results, TSC can greatly improve the clustering performance by effectively using auxiliary unlabeled data when compared with other state-of-the-art clustering algorithms.

Keywords: Transfer Learning, Spectral Clustering, Co-clustering.

1 Introduction

Clustering aims at finding groups of objects so that the objects in the same group are relatively similar while the objects in different group are relatively dissimilar. In the past decades, many clustering algorithms have been proposed, such as k-means clustering [1], spectral clustering [2, 3], Bregman divergence based clustering [4], etc. Focused on improving the clustering performance, prior knowledge in the form of must-link or cannot-link constraints [5] and auxiliary labeled data [6] have been introduced in clustering. Recently, auxiliary unlabeled data were also used to improve the performance of clustering by the so-called self-taught clustering (STC) [7]. To the best of our knowledge, STC is the first method that transfers knowledge from unlabeled data to facilitate more effective clustering. Its merit has been demonstrated in image clustering tasks.

Spectral clustering algorithms [2, 3] are well-known methods that use manifold information contained in the sample distribution to carry out the clustering task and very often outperform the traditional clustering algorithms such as the k-means algorithm. However, it seems that how to transfer knowledge from auxiliary unlabeled dataset to facilitate more effective spectral clustering has not yet been explored. In this paper, a transfer spectral clustering (TSC) algorithm is proposed and it works on the assumption that the related tasks share the same

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 789–803, 2012.

low dimensional feature embedding. This assumption could be measured by the objective of bipartite graph co-clustering. The proposed algorithm involves not only the data manifold information of the individual task but also the feature manifold information shared between tasks. The experimental results show that TSC can greatly improve the clustering performance by effectively using auxiliary unlabeled data.

This work is presented as follows. In section 2, the related works are firstly highlighted. The formulation of our method is given in section 3 which also describes the corresponding optimization method. In section 4, the experimental results are reported. In section 5, we give the conclusions and discuss the future works.

2 Related Works

Our method is related to co-clustering, transfer learning and multitask clustering with the corresponding related works highlighted below.

2.1 Co-clustering

Co-clustering aims at performing clustering on both the samples and the attributes so that clustering of attributes can help improve the clustering quality of samples. Many co-clustering algorithms have been proposed, e.g. information theoretic co-clustering (ITCC) [8] and bipartite graph co-clustering [9]. ITCC is a co-clustering algorithm based on information theory, which attempts to find the co-clusters such that the mutual information between the clustered random variables is maximized subject to constraints on the number of row and column clusters. As a totally different method, co-clustering on bipartite graph [9] expresses samples and attributes by a bipartite graph and seeks minimum cuts on this graph such that samples and attributes are both well grouped. A bipartite graph based co-clustering will be exploited by our new method to be presented in the next section.

2.2 Transfer Learning

Transfer learning [10–12] attempts to improve the learning performance on a target dataset by utilizing auxiliary datasets. It has been proved to be beneficial in practice [13, 14]. Many works have been done on inductive transfer learning [15] and transductive transfer learning [16–18].

For clustering problems, Dai et al. [7] proposed self-taught clustering (STC) to cluster a small collection of target data with the help of a large amount of unlabeled auxiliary data. STC extends the information theoretic co-clustering algorithm (ITCC) [8] with the assumption that target dataset and auxiliary dataset share the same feature clustering. STC minimizes the same loss with ITCC for the two datasets simultaneously. In this paper, we propose a method based on a similar assumption. However, unlike STC which built on information theory, our method is built on graphs.

2.3 Multitask Clustering

Multitask learning [19–21] performs multiple learning tasks concurrently to improve the individual performance. But almost all existing works have been focused on supervised settings. Recently, multitask Bregman clustering (MBC) [22] was proposed to extend Bregman divergence based clustering [4] to multitask settings. MBC aims at minimizing a local loss function for each single task and carries out a task regularization involving all tasks. The task regularization of MBC is indeed the sum of divergence between two learned density models for any pair of different tasks.

3 Transfer Spectral Clustering

3.1 Problem Formulation

Let us first describe the clustering tasks for our transfer spectral clustering (TSC) algorithm. Given two data matrix $X^{(1)} = [x_1^{(1)}, \cdots, x_{n_1}^{(1)}]$ and $X^{(2)} = [x_1^{(2)}, \cdots, x_{n_2}^{(2)}]$ containing n_1 and n_2 samples with n_f features, where $x_i^{(1)} \in R^{n_f \times n_1}$ and $x_j^{(2)} \in R^{n_f \times n_2}$, TSC aims at performing clustering on both datasets simultaneously and achieving better performance than clustering them separately. Assume that the number of clusters on both datasets is the same, and is denoted as k. To achieve better performance, we try to find low dimensional embeddings for these two datasets so that they not only are smooth on the data manifold, like that in spectral clustering [2], but also maximize the task relationship measured by co-clustering objective. With the low dimensional embeddings obtained, the traditional clustering algorithms could be applied to get the partitions for both datasets.

Let $G^{(1)}$ and $G^{(2)}$ denote the k-nn graphs constructed for each task separately. The corresponding affinity matrices $W^{(1)}$ and $W^{(2)}$ are defined as

$$W_{ij}^{(1)} = \begin{cases} 1 \text{ if } x_i^{(1)} \in \mathcal{N}^{(1)}(x_j^{(1)}) \text{ or } x_j^{(1)} \in \mathcal{N}^{(1)}(x_i^{(1)}) \\ 0 \qquad\qquad\qquad\qquad \text{otherwise,} \end{cases} \tag{1}$$

$$W_{ij}^{(2)} = \begin{cases} 1 \text{ if } x_i^{(2)} \in \mathcal{N}^{(2)}(x_j^{(2)}) \text{ or } x_j^{(2)} \in \mathcal{N}^{(2)}(x_i^{(2)}) \\ 0 \qquad\qquad\qquad\qquad \text{otherwise,} \end{cases} \tag{2}$$

where $\mathcal{N}^{(1)}(x_j^{(1)})$ and $\mathcal{N}^{(2)}(x_j^{(2)})$ denote the set of neighbors of $x_j^{(1)}$ and $x_j^{(2)}$ respectively. The first part of our goal is to obtain embeddings that are smooth over the corresponding graph. Let $F^{(1)} \in R^{n_1 \times k}$ and $F^{(2)} \in R^{n_2 \times k}$ denote the embeddings, where each row is an embedding for the corresponding sample. The smoothness of $F^{(1)}$ on $G^{(1)}$ can be measured by

$$\frac{1}{2} \sum_{i,j=1}^{n_1} W_{ij}^{(1)} \| \frac{1}{\sqrt{D_{ii}^{(1)}}} F_i^{(1)} - \frac{1}{\sqrt{D_{jj}^{(1)}}} F_j^{(1)} \|^2, \tag{3}$$

where $F_i^{(1)}$ is the ith row of $F^{(1)}$ and $D^{(1)} = diag(W^{(1)}\mathbf{1})$. It can be easily shown that equation (3) can be simplified as

$$\mathrm{tr}({F^{(1)}}^T(I - W_N^{(1)})F^{(1)}),$$

where

$$W_N^{(1)} = {D^{(1)}}^{-\frac{1}{2}} W^{(1)} {D^{(1)}}^{-\frac{1}{2}}.$$

If the constraint ${F^{(1)}}^T F^{(1)} = I$ is added, the smoothness can then be measured by

$$\mathrm{tr}({F^{(1)}}^T W_N^{(1)} F^{(1)}). \tag{4}$$

Larger value means the embeddings being smoother on the graph. Similarly, for $F^{(2)}$, the smoothness on $G^{(2)}$ can be measured by

$$\mathrm{tr}({F^{(2)}}^T W_N^{(2)} F^{(2)}), \tag{5}$$

where

$$W_N^{(2)} = {D^{(2)}}^{-\frac{1}{2}} W^{(2)} {D^{(2)}}^{-\frac{1}{2}}$$

and

$$D^{(2)} = diag(W^{(2)}\mathbf{1}).$$

In addition to pursuing smoothness on the corresponding graphs, TSC also aims at linking the two embeddings together such that one embedding obtained in one data set facilitates the finding of the embedding in another dataset. This goal can be achieved by graph co-clustering as follows.

Recall that the purpose of graph based co-clustering [9] is to find minimum cuts on a bipartite graph whose nodes include samples and features. Given the data matrix $A \in R^{d\times n}$, where d is the number of features and n is the number of samples. The affinity matrix for bipartite graph is defined as

$$W = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

and the corresponding graph Laplacian is defined as

$$L = \begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix}$$

where $D_1 = diag(A\mathbf{1})$ and $D_2 = diag(A^T\mathbf{1})$. The objective function of co-clustering [9] could be expressed as

$$\begin{aligned} \min_z \ & z^T L z \\ \text{s.t. } & x^T D_1 x = 1, \\ & y^T D_2 y = 1, \end{aligned} \tag{6}$$

where

$$z = \begin{bmatrix} x \\ y \end{bmatrix}$$

with vector x containing the embeddings for features, and vector y containing the embeddings for samples. It can be easily shown that such problem is equivalent to

$$\begin{aligned} \max_{x,y} \; & x^T A y \\ \text{s.t.} \; & x^T D_1 x = 1, \\ & y^T D_2 y = 1. \end{aligned} \tag{7}$$

Motivated by graph co-clustering, the second part of our objective function can be defined as

$$\begin{aligned} & \Omega(F^{(1)}, F^{(2)}, F^{(3)}) \\ & = \mathrm{tr}({F^{(3)}}^T X^{(1)} F^{(1)}) + \mathrm{tr}({F^{(3)}}^T X^{(2)} F^{(2)}), \end{aligned} \tag{8}$$

where the rows of matrix $F^{(3)}$ are the embeddings for features. Larger value of the above equation means better co-clustering on both datasets which share the same feature clustering. In our TSC algorithm with this measurement, we only consider normalized data matrices which are defined as

$$X_N^{(1)} = {D_1^{(1)}}^{-\frac{1}{2}} X^{(1)} {D_2^{(1)}}^{-\frac{1}{2}}, \tag{9}$$

$$X_N^{(2)} = {D_1^{(2)}}^{-\frac{1}{2}} X^{(2)} {D_2^{(2)}}^{-\frac{1}{2}}, \tag{10}$$

where

$$\begin{aligned} D_1^{(1)} &= diag(X^{(1)} \mathbf{1}), \\ D_2^{(1)} &= diag({X^{(1)}}^T \mathbf{1}), \\ D_1^{(2)} &= diag(X^{(2)} \mathbf{1}), \\ D_2^{(2)} &= diag({X^{(2)}}^T \mathbf{1}). \end{aligned}$$

Therefore, by combining these two goals, i.e., to obtain smooth embeddings on the data manifold and to maximize the co-cluster objective, our objective function can be expressed as

$$\begin{aligned} \max_{F^{(1)}, F^{(2)}, F^{(3)}} \; & \mathrm{tr}({F^{(1)}}^T W_N^{(1)} F^{(1)}) + \mathrm{tr}({F^{(2)}}^T W_N^{(2)} F^{(2)}) + \\ & \lambda(\mathrm{tr}({F^{(3)}}^T X_N^{(1)} F^{(1)}) + \mathrm{tr}({F^{(3)}}^T X_N^{(2)} F^{(2)})) \\ \text{s.t.} \; & {F^{(1)}}^T F^{(1)} = I, \\ & {F^{(2)}}^T F^{(2)} = I, \\ & {F^{(3)}}^T F^{(3)} = I, \end{aligned} \tag{11}$$

where $\lambda > 0$ is the trade off between the smoothness on graphs and the co-clustering objective. If λ is set as 0, our method becomes the classical spectral clustering.

3.2 Another View of the Formulation

In this section, we provide another view of our formulation. Let us define a graph $\tilde{G}$ whose schematic diagram is shown in Figure 1. And the corresponding affinity matrix is defined as

$$\tilde{W} = \begin{bmatrix} W_N^{(1)} & 0 & \frac{\lambda}{2}{X_N^{(1)}}^T \\ 0 & W_N^{(2)} & \frac{\lambda}{2}{X_N^{(2)}}^T \\ \frac{\lambda}{2}X_N^{(1)} & \frac{\lambda}{2}X_N^{(2)} & 0 \end{bmatrix}.$$

We can see that the left part and right part of $\tilde{G}$ are actually $G^{(1)}$ and $G^{(2)}$ respectively, and they are linked by features. If we define

$$\tilde{F} = \begin{bmatrix} F^{(1)} \\ F^{(2)} \\ F^{(3)} \end{bmatrix},$$

it can be shown that $\mathrm{tr}(\tilde{F}^T\tilde{W}\tilde{F})$ is actually the objective function of problem (11). Hence, our formulation is to find the representation for features and samples based on this graph. Our definition is different from the graphs defined in EigenTransfer [11]. $\tilde{G}$ does not include label information and considers direct relations between samples within tasks. Moreover, one might control the quantity of knowledge transferred by the parameter λ introduced. As a result, it is more suitable for clustering tasks.

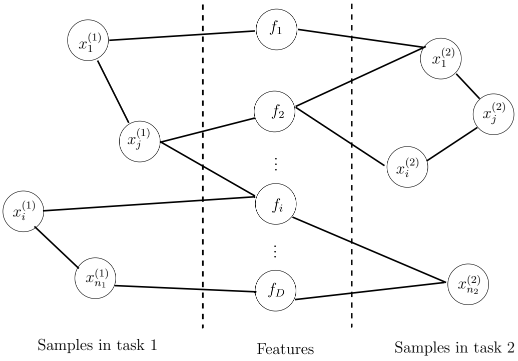


Fig. 1. Another View of TSC

3.3 Algorithm

Now, we present the algorithm to find solutions for our formulation. Problem (11) is not a convex optimization problem and hence, only a local solution can be obtained. Let us first initialize $F^{(1)}$ and $F^{(2)}$ with the top k eigenvectors of $W_N^{(1)}$ and $W_N^{(2)}$ respectively, and $F^{(3)}$ with the top k left singular vectors of the matrix normalized from

$$X = \left[X_N^{(1)} \; X_N^{(2)} \right]$$

in accordance with (9) and (10). Then, they are updated such that the value of the objective function increases until convergence is achieved.

Because of the orthonormality constraints, F_1, F_2 and F_3 actually are on the Stiefel manifold [23]. The problem (11) could be solved by repeatedly updating F_1, F_2 and F_3 on the manifold alternatively. During iterations, if F_1, F_2 and F_3 are not on the manifold, they are updated with their projections on the manifold.

For a rank p matrix $Z \in R^{n \times p}$, the projection of Z on Stiefel manifold is defined as

$$\pi(Z) = \arg \min_{Q^T Q = I} \|Z - Q\|_F^2. \tag{12}$$

If the singular value decomposition (SVD) of Z is $Z = U \Sigma V^T$, the projection could be computed as $\pi(Z) = U I_{n,p} V^T$ [23]. Hence, we can update F_1, F_2 and F_3 by moving them in the direction of increasing the value of the objective function. For convenience of descriptions, we denote the objective function as

$$\begin{aligned} & g(F^{(1)}, F^{(2)}, F^{(3)}) \\ =& \mathrm{tr}({F^{(1)}}^T W_N^{(1)} F^{(1)}) + \mathrm{tr}({F^{(2)}}^T W_N^{(2)} F^{(2)}) + \\ & \lambda(\mathrm{tr}({F^{(3)}}^T X_N^{(1)} F^{(1)}) + \mathrm{tr}({F^{(3)}}^T X_N^{(2)} F^{(2)})). \end{aligned} \tag{13}$$

Hence, the partial derivatives of g with respect to $F^{(1)}$, $F^{(2)}$ and $F^{(3)}$ can be computed as

$$\frac{\partial g}{\partial F^{(1)}} = 2W_N^{(1)} F^{(1)} + \lambda {X_N^{(1)}}^T F^{(3)}, \tag{14}$$

$$\frac{\partial g}{\partial F^{(2)}} = 2W_N^{(2)} F^{(2)} + \lambda {X_N^{(2)}}^T F^{(3)}, \tag{15}$$

$$\frac{\partial g}{\partial F^{(3)}} = \lambda X_N^{(1)} F^{(1)} + \lambda X_N^{(2)} F^{(2)}. \tag{16}$$

The steps of our TSC algorithm are summarized in Table 1.

3.4 Time Complexity

The computational cost of the proposed clustering algorithm can be analyzed as follows. As exact SVD of a $m \times n$ matrix has time complexity $O(\min\{mn^2, m^2 n\})$,

Algorithm 1. Transfer Spectral Clustering

```
Input: $W_N^{(1)}$, $W_N^{(2)}$, $X_N^{(1)}$, $X_N^{(2)}$, $k$, $\lambda$ and step length $t$
Initialize $F^{(1)}$, $F^{(2)}$ and $F^{(3)}$.
repeat
    Set $F^{(1)} = \pi(F^{(1)} + t\frac{\partial g}{\partial F^{(1)}}/\|\frac{\partial g}{\partial F^{(1)}}\|)$.
    Set $F^{(2)} = \pi(F^{(2)} + t\frac{\partial g}{\partial F^{(2)}}/\|\frac{\partial g}{\partial F^{(2)}}\|)$.
    Set $F^{(3)} = \pi(F^{(3)} + t\frac{\partial g}{\partial F^{(3)}}/\|\frac{\partial g}{\partial F^{(3)}}\|)$.
until Converge
Form matrix $Z^{(1)}$ and $Z^{(2)}$ by normalizing each rows of $F^{(1)}$ and $F^{(2)}$ to have unit length.
Treat each row of $Z^{(1)}$ and $Z^{(2)}$ as samples and run $k$-means to get partitions $P^{(1)}$ and $P^{(2)}$.
Output: $P^{(1)}$ and $P^{(2)}$.
```

the initialization step, which actually involves eigenvalue decomposition and SVD, has time complexity $O(n_1^2k+n_2^2k+\min\{D^2(n_1+n_2), D(n_1+n_2)^2\})$. During iterations, TSC computes the new embeddings and projections. The computations of all three new embeddings involve matrix products and additions and the time complexity is $O(n_1^2k + n_2^2k + D^2k)$. To find projections, computing SVD of matrices F_1, F_2 and F_3 are needed. As such, the time complexity for computing new embeddings and projections is $O(n_1^2k + n_2^2k + D^2k)$. Thus, the time complexity of the iteration part is $O(iter(n_1^2k + n_2^2k + D^2k))$, where *iter* is the number of iterations. Since k is usually small compared with the number of features and samples, our method is computationally efficient.

4 Experimental Results

In this section, an evaluation of the TSC algorithm on text clustering tasks and a comparison with spectral clustering (SC) [2], self-taught clustering (STC) [7] and multitask Bregman clustering (MBC) [22] are reported.

4.1 Datasets

We test our algorithms on clustering tasks generated from the 20 Newsgroups (20NG) dataset[1] and Reuters-21578 data set[2]. The 20NG dataset used in our experiments is the bydate version. This version is sorted by date into training(60%) and test(40%) sets. After preprocessing, the 20NG dataset contains 18774 documents and 61188 terms and our datasets have been extracted from it as follows. Each dataset contains two separate parts, with each part corresponding to a clustering task. The categories selected for these datasets are listed in Table 1. The second column of Table 1 is for task 1 while the third column is for task

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/

2. We can see that the subcategories of tasks 1 and 2 of the first 2 data sets are different, but belong to a more general category. We select 500 samples for each class. For datasets 20NG3-10, the subcategories of the two tasks are the same and we use the whole test set in task 2 and select the same number of samples from the training set in task 1. Hence, samples of these two tasks follow different distribution. For the 20NG11-16 datasets, the samples in the two tasks share the same top categories, but they are selected from different subcategories. Therefore, the distributions of samples in these two tasks are different. For each class, we also select 500 samples. Reuters1-3 were generated in a similar way as 20NG11-16, but with all the samples used in our experiments. All datasets are represented in *tf-idf* format and normalized such that each sample has unit length.

Table 1. Datasets Generated from 20 Newsgroups and Reuters-21578 Datasets

Datasets	Task 1	Task 2
20NG1	sci.crypt sci.electronics	sci.med sci.space
20NG2	talk.politics.guns talk.politics.mideast	talk.politics.misc talk.religion.misc
20NG3	comp.graphics comp.os.ms-windows.misc	comp.graphics comp.os.ms-windows.misc
20NG4	rec.autos rec.motorcycles	rec.autos rec.motorcycles
20NG5	sci.crypt sci.electronics	sci.crypt sci.electronics
20NG6	talk.politics.guns talk.religion.misc	talk.politics.guns talk.religion.misc
20NG7	alt.atheism comp.graphics	alt.atheism comp.graphics
20NG8	misc.forsale rec.sport.hockey	misc.forsale rec.sport.hockey
20NG9	rec.sport.hockey sci.electronics	rec.sport.hockey sci.electronics
20NG10	sci.space soc.religion.christian	sci.space soc.religion.christian
20NG11	comp.*, rec.*	comp.*, rec.*
20NG12	comp.*, sci.*	comp.*, sci.*
20NG13	comp.*, talk.*	comp.*, talk.*
20NG14	rec.*, sci.*	rec.*, sci.*
20NG15	rec.*, talk.*	rec.*, talk.*
20NG16	sci.*, talk.*	sci.*, talk.*
Reuters1	orgs.*, places.*	orgs.*, places.*
Reuters2	people.*, places.*	people.*, places.*
Reuters3	orgs.*, people.*	orgs.*, people.*

4.2 Clustering Performance Measures

In order to measure the clustering performance, the normalized mutual information (NMI), cluster purity (purity) and rand index (RI) [24] were adopted. For $\Omega = \{\omega_1, \cdots, \omega_K\}$ denoting a set of clusters and $\mathbb{C} = \{c_1, \cdots, c_K\}$ referring to a set of classes, where ω_i is interpreted as the samples in cluster i, and c_j as the set of samples in class j, these three measures are defined as

$$NMI = \frac{\sum_{jl} n_{jl} \log\left(\frac{N n_{jl}}{n_j n_l}\right)}{\sqrt{(\sum_j n_j \log \frac{n_j}{N})((\sum_l n_l \log \frac{n_l}{N})}}, \tag{17}$$

$$purity = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|, \tag{18}$$

$$RI = \frac{TP + TN}{TP + FP + FN + TN}, \tag{19}$$

where n_j, n_l are the numbers of samples in class j and in cluster l, respectively, n_{jl} is the number of samples occurring in both class j and cluster l, and $TN+TP$ is the number of correct decisions for a pair of objects, $FP+FN$ is the number of wrong decisions. RI computes the percentage of correct pair-wise relationships. For all these three measures, larger value means better cluster performance. For all the methods in our evaluation, we compute the average performance of the two tasks.

4.3 Empirical Analysis

In all the experiments, we construct k-nn graphs with $k = 27$. We apply spectral clustering on data from individual task (SC$_{ind}$) and on data combined from all tasks (SC$_{com}$). For SC$_{com}$, we find the embeddings for all tasks and perform k-means within tasks. For self-taught clustering (STC), we set λ as 1, because the two tasks are equally important in all our experiments. The number of feature clusters is set as 32 as in [7]. The maximum number of iterations for STC is set as 20. MBC and STC are initialized with spectral clustering which usually achieves better performance than k-means. The Bregman divergence used in MBC is the Euclidean distance and the only parameter for MBC is set as 0.5 as in [22]. For our method, we set $\lambda = 3$, and the step length as 1. In each experiment, we run the k-means algorithm 100 times with random starting points and the most frequent cluster assignment is used. Spectral clustering is not designed for more than one clustering tasks. Hence, we perform spectral clustering on each task separately and report the average performance.

In Table 2 and Table 3, the clustering performance of the five methods on these datasets is reported. We can see that our method achieves the best performance in most datasets. It can be explained by the fact that the feature embeddings computed by our method have taken into considerations of the

Table 2. Clustering performances (row 1: purity, row 2: NMI, row 3: RI) of different algorithms on the first ten datasets in Table 1

Datasets	SC$_{ind}$	SC$_{com}$	STC	MBC	TSC
20NG1	**0.9605**	0.951	0.9295	0.941	0.955
	0.7643	0.718	0.6339	0.7201	0.7405
	0.9243	0.9068	0.8692	0.8894	0.9141
20NG2	0.9425	0.8495	0.9035	**0.9445**	0.94
	0.6894	0.5175	0.5743	**0.7048**	0.6883
	0.8923	0.7701	0.8311	**0.8962**	0.8894
20NG3	0.7314	0.8237	0.7308	0.7603	**0.8295**
	0.1822	0.3324	0.1685	0.2261	**0.3497**
	0.6092	0.7108	0.6067	0.6355	**0.7184**
20NG4	0.8643	0.8883	0.7841	0.8883	**0.904**
	0.4615	0.5294	0.2788	0.5464	**0.5679**
	0.7671	0.8017	0.6623	0.8025	**0.8265**
20NG5	0.9353	**0.948**	0.8934	0.9194	0.9416
	0.6725	**0.7166**	0.5196	0.6307	0.6961
	0.8805	**0.9024**	0.8107	0.8529	0.8919
20NG6	0.9114	0.935	0.9	**0.939**	0.935
	0.5776	0.668	0.5255	**0.679**	0.6687
	0.8422	0.879	0.8208	**0.8873**	0.8797
20NG7	0.9745	**0.9788**	0.9668	0.9724	0.9781
	0.835	**0.8613**	0.791	0.818	0.8567
	0.9506	**0.9589**	0.9359	0.9464	0.9573
20NG8	0.9571	0.9654	0.9597	0.9558	**0.9731**
	0.7746	0.7951	0.766	0.7451	**0.8359**
	0.9185	0.9334	0.9227	0.9159	**0.9478**
20NG9	0.9842	**0.9855**	0.9754	0.971	**0.9855**
	0.8849	0.8927	0.8441	0.835	**0.8964**
	0.969	**0.9714**	0.952	0.9436	**0.9714**
20NG10	0.9728	0.9791	0.9715	0.9665	**0.9835**
	0.825	0.8549	0.8155	0.8108	**0.8815**
	0.9471	0.9591	0.9447	0.9352	**0.9677**

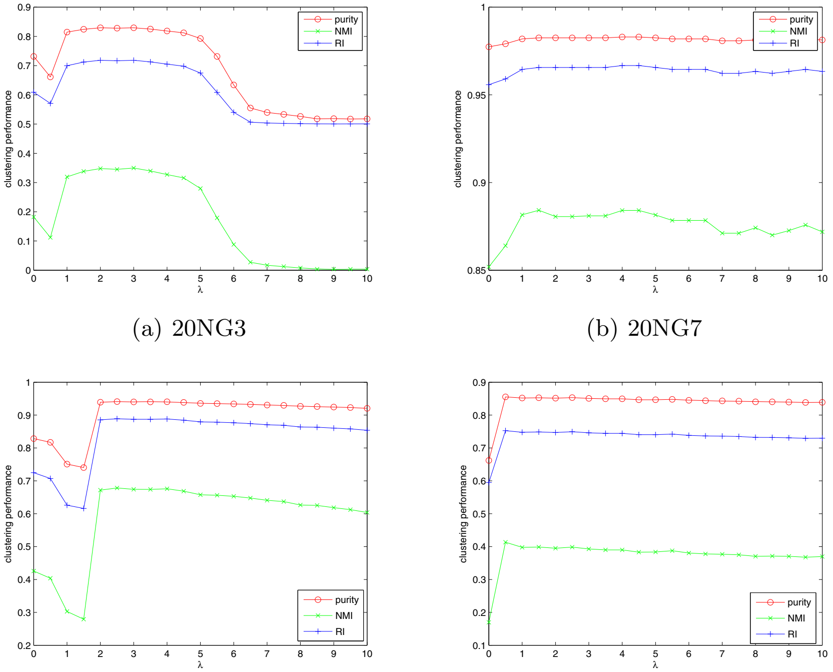
embeddings of samples in the two clustering tasks and such a sharing could improve the clustering performance in quite a significant manner.

In the proposed TSC algorithm, the parameter λ controls the quantity of information to be transferred and a sensitivity analysis of this parameter has been carried out. Figure 2 presents the clustering performance with respect to different λ on datasets 20NG3, 20NG7, 20NG16, and Reuters1. We can see that the high performance of our algorithm is stable in a range of λ. The performance first goes up when λ is small, and drops when λ is too big. Thus, a proper amount of knowledge should be transfered and we set $\lambda = 3$ in all our experiments.

Table 3. Clustering performances (row 1: purity, row 2: NMI, row 3: RI) of different algorithms on the last nine datasets in Table 1

Datasets	SC_{ind}	SC_{com}	STC	MBC	TSC
20NG11	0.9785	**0.979**	0.947	0.9725	0.9745
	0.8531	**0.8553**	0.7131	0.8217	0.8323
	0.9581	**0.9589**	0.9007	0.9466	0.9505
20NG12	0.867	0.851	0.8385	0.853	**0.8875**
	0.522	0.4916	0.4176	0.485	**0.5304**
	0.7825	0.7619	0.74	0.7605	**0.806**
20NG13	0.9785	0.975	0.9735	0.977	**0.9845**
	0.8507	0.8332	0.8243	0.8475	**0.8856**
	0.9579	0.9513	0.9484	0.9551	**0.9695**
20NG14	0.89	0.8755	0.868	0.8815	**0.922**
	0.5918	0.5641	0.4835	0.5953	**0.6452**
	0.8183	0.7973	0.7767	0.8082	**0.8608**
20NG15	0.8695	0.849	0.861	0.874	**0.9605**
	0.6049	0.5758	0.553	0.5625	**0.7838**
	0.7975	0.7764	0.784	0.7852	**0.9247**
20NG16	0.8285	0.8035	0.815	0.831	**0.94**
	0.4257	0.3876	0.4019	0.4202	**0.6742**
	0.7249	0.722	0.7085	0.7203	**0.8872**
Reuters1	0.6621	0.6779	0.6675	0.8313	**0.8506**
	0.1697	0.1861	0.1612	0.349	**0.3929**
	0.5949	0.6098	0.594	0.7201	**0.746**
Reuters2	0.72	0.7408	0.7387	0.7388	**0.7647**
	0.1775	0.182	0.1806	0.2102	**0.2196**
	0.6037	0.616	0.6162	0.6177	**0.6408**
Reuters3	0.6356	**0.6903**	0.6476	0.6253	0.6527
	0.0706	**0.0921**	0.0734	0.0394	0.0784
	0.5429	**0.5727**	0.55	0.5303	0.5531

The convergence property of TSC was also studied. As the objective function of TSC is a continuous function of $F^{(1)}$, $F^{(2)}$ and $F^{(3)}$, its value will increase monotonically as long as the step size t is small enough. Hence, TSC will converge in a fixed number of iterations. Moreover, the required number of iterations to converge depends on the threshold required for checking convenience. Usually, the smaller the threshold, the more iterations are needed. In our experiments, we use 0.00001, and the number of iterations to converge is less than a hundred. Figure 3 shows the learning curve of TSC in dataset Reuters1. We can see that TSC converges very well in this dataset.



(a) 20NG3

(b) 20NG7

(c) 20NG16

(d) Reuters1

Fig. 2. Clustering performance with different λ

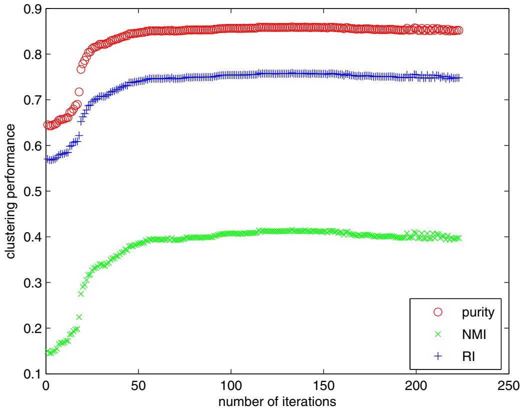


Fig. 3. The learning curve of the TSC algorithm in dataset Reuters1

5 Conclusions and Future Work

In this paper, an approach to transfer knowledge from other datasets to facilitate more effective spectral clustering is presented. Our assumption is that embeddings of features could link different clustering tasks (from different datasets) and hence improve the clustering performance with respect to each other. Based on this assumption, a co-clustering objective is proposed to design a new spectral clustering algorithm called transfer spectral clustering (TSC). The experimental results show that with the help of co-clustering, TSC has outperformed several state-of-the-art clustering algorithms.

The proposed TSC algorithm assumes that the two datasets have the same number of clusters, which limits its applications in practice. In the future, we plan to extend our method to allow clustering datasets into different groups. We will also explore other more efficient optimization method for TSC. In addition, we will try to find a way to tune the parameter λ so that the quantity of knowledge transferred can be made automatic.

Acknowledgment. We want to thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the Hong Kong Polytechnic University under Research Studentship.

References

1. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Inc., Upper Saddle River (1988)
2. Ng, A.Y., Jordan, M.I., Weiss, Y.: On Spectral Clustering: Analysis and an algorithm, pp. 849–856 (2001)
3. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 22(8), 888–905 (2000)
4. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with Bregman Divergences. J. Mach. Learn. Res. 6, 1705–1749 (2005)
5. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained K-means Clustering with Background Knowledge. In: Proceedings of the Eighteenth International Conference on Machine Learning, ICML 2001, pp. 577–584. Morgan Kaufmann Publishers Inc., San Francisco (2001)
6. Finley, T., Joachims, T.: Supervised clustering with support vector machines. In: Proceedings of the 22nd International Conference on Machine Learning, ICML 2005, pp. 217–224. ACM, New York (2005)
7. Dai, W., Yang, Q., Xue, G.-R., Yu, Y.: Self-taught clustering. In: Proceedings of the 25th International Conference on Machine Learning, ICML 2008, pp. 200–207. ACM, New York (2008)
8. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 89–98. ACM, New York (2003)
9. Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001, pp. 269–274. ACM, New York (2001)

10. Pan, S.J., Yang, Q.: A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22, 1345–1359 (2010)
11. Dai, W., Jin, O., Xue, G.R., Yang, Q., Yu, Y.: EigenTransfer: a unified framework for transfer learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 193–200. ACM, New York (2009)
12. Mahmud, M.M., Ray, S.: Transfer Learning using Kolmogorov Complexity: Basic Theory and Empirical Evaluations. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems 20, pp. 985–992. MIT Press, Cambridge (2008)
13. Pan, S.J., Zheng, V.W., Yang, Q., Hu, D.H.: Transfer learning for WiFi-based indoor localization. In: Proceedings of the Workshop on Transfer Learning for Complex Task of the 23rd AAAI Conference on Artificial Intelligence (2008)
14. Blitzer, J., Dredze, M., Pereira, F.: Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic, pp. 440–447. Association for Computational Linguistics (June 2007)
15. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th International Conference on Machine Learning, ICML 2007, pp. 759–766. ACM, New York (2007)
16. Daumé, I.H., Marcu, D.: Domain adaptation for statistical classifiers. J. Artif. Int. Res. 26, 101–126 (2006)
17. Zadrozny, B.: Learning and evaluating classifiers under sample selection bias. In: Proceedings of the Twenty-first International Conference on Machine Learning, ICML 2004, pp. 903–910. ACM, New York (2004)
18. Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of Statistical Planning and Inference 90(2), 227–244 (2000)
19. Caruana, R.: Multitask Learning. Machine Learning 28, 41–75 (1997)
20. Argyriou, A., Micchelli, C.A., Pontil, M., Ying, Y.: A Spectral Regularization Framework for Multi-Task Structure Learning. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems 20, pp. 25–32. MIT Press, Cambridge (2008)
21. Ando, R.K., Zhang, T.: A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. J. Mach. Learn. Res. 6, 1817–1853 (2005)
22. Zhang, J., Zhang, C.: Multitask Bregman Clustering. In: Fox, M., Poole, D. (eds.) Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, pp. 655–660. AAAI Press (2010)
23. Manton, J.H.: Optimization algorithms exploiting unitary constraints. IEEE Transactions on Signal Processing 50(3), 635–650 (2002)
24. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)

An Aspect-Lexicon Creation and Evaluation Tool for Sentiment Analysis Researchers*

Mus'ab Husaini[1], Ahmet Koçyiğit[1], Dilek Tapucu[1,2],
Berrin Yanikoglu[1], and Yücel Saygın[1]

[1] Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey
{musabhusaini,akocyigit,dilektapucu,berrin,ysaygin}@sabanciuniv.edu
[2] Dept. of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey

Abstract. In this demo paper, we present SARE, a modular and extendable semi-automatic system that *1*) assists researchers in building gold-standard lexicons and evaluating their lexicon extraction algorithms; and *2*) provides a general and extendable sentiment analysis environment to help researchers analyze the behavior and errors of a core sentiment analysis engine using a particular lexicon.

1 Introduction

In sentiment analysis, domain aspect (also called feature) extraction is crucial for gaining a deeper and more refined understanding of the opinions expressed in a given document [3,2]. Without domain-specific aspects, the sentiment analysis process remains prone to generalizations and dilution of opinions. A *domain aspect lexicon* consists of a set of *aspects* that are broad features of the domain; and for each aspect, a set of aspect-related *keywords*. For example, in the *hotel* domain, "room quality" might be one such aspect and the terms "furniture" and "size" could be keywords associated with this aspect.

Several automatic and semi-automatic methods have been proposed in the literature to extract a domain aspect lexicon from a given domain corpus, such as those cited in [5]. In evaluating their methods, researchers either compare the coverage of the extracted lexicon to that of a hand-built one considered to be the gold standard; or they compare the performance of a baseline sentiment analysis system using the generated lexicon versus some other available lexica. The gold-standard lexicon mentioned in the former case is obtained through one of the following ways: *a*) by manually tagging words from a domain corpus; *b*) by one or more domain experts choosing aspects and keywords without the use of a corpus; or *c*) using review sets that have already been annotated with aspects and keywords by the original reviewers. The first approach is naturally rather tedious as domain corpora are usually too large to be manually processed. The second one is vulnerable to generalization error since the experts' vocabulary tends to be narrower compared to the broader vocabulary of a mass of reviewers. Finally, the third approach is not always possible, since such review sets are not available

* SARE was developed in the context of UBIPOL (Ubiquitous Participation Platform for Policy Making) project funded by European Commission, FP7.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 804–807, 2012.

in all cases. It is also difficult to verify and evaluate a hand-built lexicon to make sure that it contains all the relevant words and only the relevant words. There are no tools freely available, to the best of our knowledge, that deal with all of these challenges and assist the researcher with these tasks. We developed *SARE* (Sentiment Analysis Research Environment) to fulfill this need.

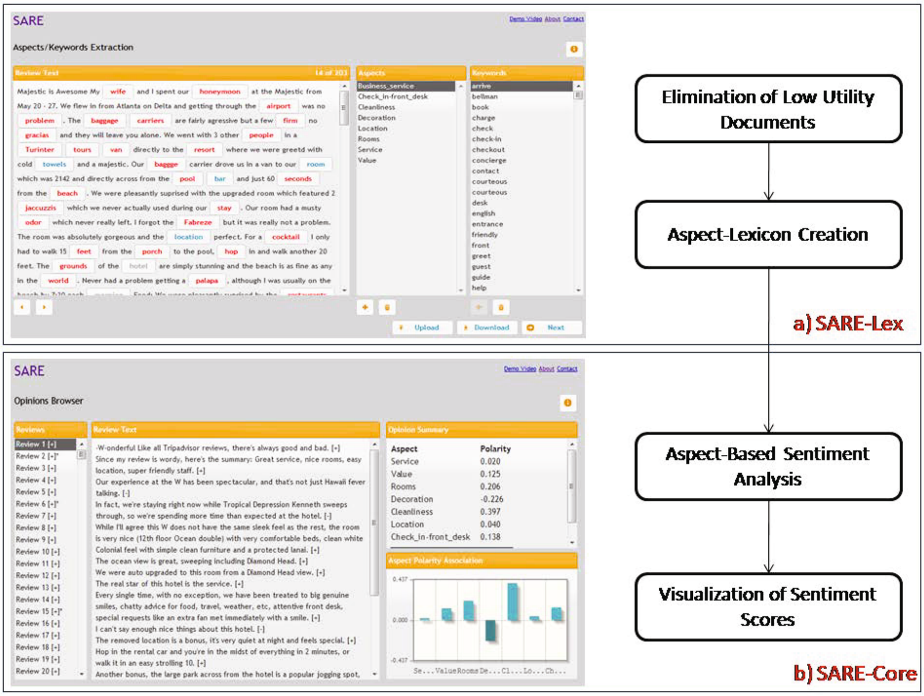
2 Overview and Main Contributions

In this demo paper, we present SARE, which is designed to support researchers in constructing and analyzing their sentiment analysis systems addressing the issues raised in Sect. 1. Currently, SARE consists of two modules: *SARE–Lex* (Lexicon Creation Tool) provides a semi-automatic method for developing gold-standard domain lexica by reducing the size of the corpus that needs to be processed manually. *SARE–Core* (Core System Results), takes a review corpus, a sentiment analysis engine, and a domain lexicon; and displays aspect-based as well as overall sentiment summary for each review in the corpus. The second module can take as input a lexicon generated by the first or a pre-generated lexicon obtained as output from a separate feature-extraction algorithm. In either case, the goal is to help the researcher analyze the behavior and errors of a core sentiment analysis engine using a particular lexicon. Thus, while SARE is intended to be a general sentiment analysis research environment, its two modules can be used in creating a hand-built lexicon and assessing the performance of any given lexicon.

Our main contribution is a publicly-available and open-source tool for producing gold-standard domain aspect lexica. This tool is primarily intended for use by sentiment analysis researchers, and to a certain extent, practitioners. For demonstrating SARE, we will present an interactive scenario for creating a lexicon from scratch given a corpus; then we will show the capabilities of the system for analyzing the output of the sentiment analysis module. Our future work involves extending SARE, which is designed to be highly modular and extendible, to support research in various sentiment analysis sub-problems by introducing pluggable components such as custom sentiment analysis engines and polarity lexica. In this work, we only deal with features that are explicitly expressed using nouns such as the ones mentioned above; the task of extracting implicit features such as those expressed using adjectives and adverbs is left for future work. We also plan on extending the system to add support for languages other than English which only requires substituting the NLP process and polarity lexicon.

3 Process and Modules

SARE is a self-contained web application that can be deployed to any Java-based web server. The system allows for a domain corpus to be imported in a variety of formats including text and XML. This corpus is then analyzed using the method described in Sect. 3.1 to obtain the most informative documents from which the user extracts domain aspects and related keywords to create the aspect lexicon. Alternatively, a pre-generated lexicon can be uploaded by the user as well. The lexicon, manually extracted or otherwise, is


SARE
Demo Video About Contact
Elimination of Low Utility Documents
Aspect-Lexicon Creation
a) SARE-Lex
Aspect-Based Sentiment Analysis
Visualization of Sentiment Scores
b) SARE-Core
Upload
Download
Next
0.437
0.000
-0.437
Se... Value Rooms De... Cl... Lo... Ch...

SARE

Aspects/Keywords Extraction

Review Text 14 of 203

Majestic is Awesome My wife and I spent our honeymoon at the Majestic from May 20 - 27. We flew in from Atlanta on Delta and getting through the airport was no problem . The baggage carriers are fairly agressive but a few firm no gracias and they will leave you alone. We went with 3 other people in a Turinter tours van directly to the resort where we were greetd with cold towels and a majestic. Our baggge carrier drove us in a van to our room which was 2142 and directly across from the pool bar and just 60 seconds from the beach . We were pleasantly suprised with the upgraded room which featured 2 jaccuzzis which we never actually used during our stay . Our room had a musty odor which never really left. I forgot the Fabreze but it was really not a problem. The room was absolutely gorgeous and the location perfect. For a cocktail I only had to walk 15 feet from the porch to the pool, hop in and walk another 20 feet. The grounds of the hotel are simply stunning and the beach is as fine as any in the world . Never had a problem getting a palapa , although I was usually on the

Aspects
- Business_service
- Check_in-front_desk
- Cleanliness
- Decoration
- Location
- Rooms
- Service
- Value

Keywords
- arrive
- bellman
- book
- charge
- check
- check-in
- checkout
- concierge
- contact
- courteous
- courteous
- desk
- english
- entrance
- friendly
- front
- greet
- guest
- guide
- help

SARE

Opinions Browser

Reviews
- Review 1 [+]
- Review 2 [+]*
- Review 3 [+]
- Review 4 [+]
- Review 5 [+]
- Review 6 [+]*
- Review 7 [+]
- Review 8 [+]
- Review 9 [+]
- Review 10 [+]
- Review 11 [+]
- Review 12 [+]
- Review 13 [+]
- Review 14 [+]
- Review 15 [+]*
- Review 16 [+]
- Review 17 [+]
- Review 18 [+]
- Review 19 [+]
- Review 20 [+]

Review Text

-W-onderful Like all Tripadvisor reviews, there's always good and bad. [+]
Since my review is wordy, here's the summary: Great service, nice rooms, easy location, super friendly staff. [+]
Our experience at the W has been spectacular, and that's not just Hawaii fever talking. [-]
In fact, we're staying right now while Tropical Depression Kenneth sweeps through, so we're spending more time than expected at the hotel. [-]
While I'll agree this W does not have the same sleek feel as the rest, the room is very nice (12th floor Ocean double) with very comfortable beds, clean white Colonial feel with simple clean furniture and a protected lanai. [+]
The ocean view is great, sweeping including Diamond Head. [+]
We were auto upgraded to this room from a Diamond Head view. [+]
The real star of this hotel is the service. [+]
Every single time, with no exception, we have been treated to big genuine smiles, chatty advice for food, travel, weather, etc, attentive front desk, special requests like an extra fan met immediately with a smile. [+]
I can't say enough nice things about this hotel. [-]
The removed location is a bonus, it's very quiet at night and feels special. [+]
Hop in the rental car and you're in the midst of everything in 2 minutes, or walk it in an easy strolling 10. [+]
Another bonus, the large park across from the hotel is a popular jogging spot,

Opinion Summary

Aspect	Polarity
Service	0.020
Value	0.125
Rooms	0.206
Decoration	-0.226
Cleanliness	0.397
Location	0.040
Check_in-front_desk	0.138

Aspect Polarity Association

Fig. 1. SARE: (a) SARE-Lex module (b) SARE-Core module

then used by the sentiment analysis engine to process the corpus and results are displayed to the user for further analysis. Figure 1 illustrates this workflow with the help of screenshots. The system currently uses a fixed sentiment analysis engine and polarity lexicon, but will be extended to accept pluggable versions in the near future. This application is publicly accessible online by visiting `http://ferrari.sabanciuniv.edu/sare`, and a demo video explaining its usage can be downloaded from `http://ferrari.sabanciuniv.edu:81/public/videos/sare-demo.wmv`. It should be noted that SARE is developed on the principles of REST architecture, which allows other systems to easily access it, use its output independent of the website, and extend it as desired. The list of necessary API calls for this purpose can be provided on demand. In the following subsections, we will discuss the major components of the system.

3.1 SARE-Lex: Lexicon Extraction Module

This module deals with the problem of aspect lexicon extraction by corpus summarization and user annotation. We approximate aspect keywords with corpus nouns and apply a variation of the *Greedy Set Cover* algorithm that we developed called *Eagerly Greedy Set Cover* algorithm to find the minimum set of documents that cover all of the nouns in the corpus. Conceptually, each noun is assigned a utility score equal to its corpus frequency, and the utility score of each document is calculated as the cumulative utility score of the nouns it contains. However, since we have formulated this as a set cover problem, a given noun can only lend its score to one document, and therefore we find the best placement for each noun where it maximizes document utility. For large enough corpora, this generally results in a utility distribution where most of the documents have very low utility score. By ignoring documents with utilities in the lower percentile (a percentile threshold), the number of documents requiring annotation can be significantly reduced. This threshold is a parameter of the module and the tool assists the user in selecting it by displaying a utility distribution. The reduced set of documents is then displayed sequentially to the user for collecting new aspects and associated keywords. As each document is displayed, the interface emphasizes document nouns for better visibility. The user can then interact with these emphasized nouns to mark them as aspects or keywords of an aspect.

3.2 SARE-Core: Sentiment Analysis Module

The objective of this module is to use the domain aspect lexicon from the previous module for calculating aspect-based and overall sentiment scores for each review, and present a summarized result. If reviews in the provided corpus were labeled as positive or negative, then the interface also indicates erroneous classifications for each review. To calculate sentiment scores, word polarities are first obtained from the SentiWordNet polarity lexicon [1]. A polarity-placement algorithm is then used to calculate score values for each aspect and the overall review. Using syntactic dependencies obtained through the Stanford NLP Parser [4], polarity values can be transferred from the polarity word to the aspect keyword, and consequently to the aspect. The tool also displays polarities of polar words, sentences, and other intermediate results. Currently, only the default engine can be used. As future work, we plan to introduce the option to plug in any sentiment analysis engine provided as a web service. We are also planning an extension to allow the user to provide their own polarity lexicon instead of the default one.

References

1. Esuli, A., Sebastiani, F.: Sentiwordnet: A publicly available lexical resource for opinion mining. In: Proc. of LREC, vol. 6, pp. 417–422 (2006)
2. Liu, B.: Sentiment analysis and subjectivity. In: Handbook of Natural Language Processing, pp. 627–666 (2010)
3. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(1-2), 1–135 (2008)
4. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proc. of the 2003 Conf. of the NAACL, vol. 1, pp. 173–180 (June 2003)
5. Zhai, Z., Liu, B., Xu, H., Jia, P.: Clustering product features for opinion mining. In: Proc. of the 4th ACM International WSDM Conf., pp. 347–354. ACM (2011)

Association Rule Mining Following the Web Search Paradigm

Radek Škrabal, Milan Šimůnek, Stanislav Vojíř, Andrej Hazucha,
Tomáš Marek, David Chudán, and Tomáš Kliegr

Department of Information and Knowledge Engineering, University of Economics,
Nám. Winstona Churchilla 4, Prague 3, 130 67, Czech Republic
{xskrr06,simunek,stanislav.vojir,andrej.hazucha,xmart17,
david.chudan,tomas.kliegr}@vse.cz

Abstract. I:ZI Miner (sewebar.vse.cz/izi-miner) is an association rule mining system with a user interface resembling a search engine. It brings to the web the notion of interactive pattern mining introduced by the MIME framework at ECML'11 and KDD'11. In comparison with MIME, I:ZI Miner discovers multi-valued attributes, supports the full range of logical connectives and 19 interest measures. A relevance feedback module is used to filter the rules based on previous user interactions.

1 Introduction

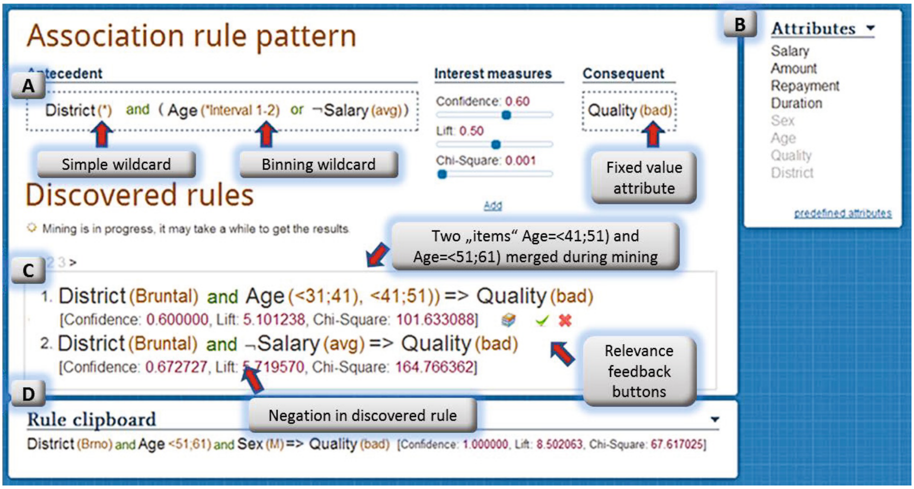
The goal of the association rule mining task is to discover patterns interesting for the user in a given collection of objects. The user interest is defined through a set of features that can appear on the left and right side of the discovered rules and by thresholds on selected interest measures. A typical task produces many rules that formally match these criteria, but only few are interesting to the user [1]. The uninteresting rules can be filtered using domain knowledge; the challenge is to balance the investment of user's time to provide the required input with the utility gained from the filtered mining result.

To address this challenge, we draw inspiration from the information retrieval task, which tackles a similar problem – select documents interesting to the user from the many that match the user's query. Web search engines are successful in retrieving subjectively interesting documents from their index; with I:ZI Miner[1] we try to apply the underlying principles to the task of discovering subjectively interesting rules from the dataset. We consider these principles to be interactivity, simplicity of user interface, immediate response and relevance feedback.

I:ZI Miner is based on similar ideas as the MIME framework [2], a desktop application introduced at ECML 2011 and KDD 2011. The main differences are that I:ZI Miner works with *multi-valued attributes* (see examples in Fig. 1) and supports the *full range of logical connectives*. Additional features that distinguish I:ZI Miner from MIME include *web interface*, *rule filtering* based on relevance feedback and a *preprocessing module*.

[1] The name I:ZI Miner is pronounced as 'easy miner'.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 808–811, 2012.



District (*) and (Age (*Interval 1-2) or ¬Salary (avg))

Confidence: 0.60

Lift: 0.50

Chi-Square: 0.001

Quality (bad)

Mining is in progress, it may take a while to get the results

1. District (Bruntal) and Age (<31;41), <41;51)) => Quality (bad)
   [Confidence: 0.600000, Lift: 5.101238, Chi-Square: 101.633088] ✓ ✖
2. District (Bruntal) and ¬Salary (avg) => Quality (bad)
   [Confidence: 0.672727, Lift: 5.719570, Chi-Square: 164.766362]

District (Brno) and Age <51;61) and Sex (M) => Quality (bad) [Confidence: 1.000000, Lift: 8.502063, Chi-Square: 67.617025]

Fig. 1. I:ZI Miner screenshot

2 User Interface

The mining task is defined in the *Pattern Pane* (Fig. 1A) by selecting interest measures and placing attributes from the *Attribute Palette* (Fig. 1B). Real-time results are shown in the *Result Pane* (Fig. 1C), which also serves for relevance feedback. Interesting rules are saved to the *Rule Clipboard* (Fig. 1D).

Pattern Pane. By dragging attributes from the Attribute palette to the antecedent and consequent of the rule, the user creates an intuitively understandable ‘rule pattern’ that discovered rules must match. Attributes are by default connected by conjunction, but it can be changed to disjunction. For a specific attribute, the user can either select a value or a wildcard (ref. to Sec. 3). There is also the option to put negation on an attribute. The user adds at least one interest measure and its threshold. A unique feature is that any combination of the available 19 interest measures can be used.

Result Pane. It displays the rules as soon as they have been discovered, i.e. the user does not have to wait for the mining process to finish to get the first results. If the discovered rule is only a confirmation [4] of a known rule, it is visually suppressed by gray font. In contrast, exception to a known rule is highlighted in red. Uninteresting rules that pass the domain knowledge check can be discarded by clicking on the red cross; this stores negative relevance feedback. Green tick moves the rule to the Rule clipboard and stores positive relevance feedback.

Attribute Palette. For an attribute to be used during mining, the user needs to drag it to the Pattern pane. If the ‘best pattern’ feature is on, the attributes are ordered according to their estimated value for predicting the consequent.

3 Architecture, Performance and Expressiveness

The software has a web service architecture; an extension of the PMML format (www.dmg.org) is used for communication between individual modules. The mining module is reused from the LISp-Miner system (lispminer.vse.cz). It is written in C++, uses a proprietary bitstring-based mining algorithm derived from the GUHA method [3], and offers a range of unique features including:

- Arbitrary combination of interest measures, including confidence, support, lift, Chi-square, Fisher and eight other interest measures developed within the GUHA method.
- Full range of logical connectives (conjunction, disjunction and negation).
- *Simple wildcard* on an attribute tells the miner to generate as many 'items' as there are values of the attribute. This is similar to other association rule mining systems that support multiple attributes.
- Adding a *Fixed value attribute* to the mining setting allows the user to limit the search space only to rules containing a selected attribute-value pair.
- *Binning wildcards* can be used to instruct the miner to dynamically merge multiple values into one 'item' during mining. If value order was specified in the preprocessing stage, only adjacent values can be binned.
- Support for distributed computing on top of the Techila platform [6].

The relevance feedback module [4] is a Java application running on top of the XML Berkeley database. I:ZI Miner is part of the SEWEBAR-CMS project [5], which provides data preprocessing and reporting capabilities.

4 Comparison with the MIME Framework

Our system shares many features with the MIME framework introduced in last year's ECML and KDD conferences. In this section we will list the features the two system share, the additional features of I:ZI Miner in comparison with MIME, and the features of MIME not implemented in I:ZI Miner, in turn.

Both systems have a 'best pattern' extension. While MIME orders *items* according to their impact on the existing mined pattern, I:ZI Miner implements a heuristic algorithm which orders *attributes* according to their estimated impact.[2] Another common feature is the ability to define groups of items which are considered as one value during mining. While in MIME these groups are defined manually, I:ZI Miner features multiple types of binning wildcards.

I:ZI Miner unique features include mining over multi-valued attributes, negation and disjunction in rules, wider choice of interest measures and filtering of discovered rules with relevance feedback.[2] Technologically, I:ZI Miner is a web application. Through integration with SEWEBAR-CMS it offers a preprocessing and reporting capability. Concerning scalability, mining runs as a web service with the underlying grid platform giving an option to upscale to Microsoft Azure.

Rule post-processing and visualization algorithms, on the other hand, are only implemented in MIME.

[2] Its technical description cannot be included for space reasons.

5 Screencasts and Demo

Screencasts and a live demonstration of I:ZI Miner on the ECML PKDD'99 Financial dataset are available at `sewebar.vse.cz/izi-miner`.

Screencast 1: Explorative Task featuring Binning Wildcards. In the exploration mode the user investigates the relationships in the dataset without being bound to a specific outcome. Unlike other association rule mining systems that operate on binary items, in I:ZI Miner the task is defined directly on multi-valued attributes (Fig. 1A). For attributes having many values with low support. I:ZI Miner offers a unique feature – binning wildcards, which allow to group fine-grained values on the fly, thus producing 'items' with higher support. Fig. 1C shows a rule with two values of the Age attribute grouped by a binning wildcard.

Screencast 2: Predictive Task featuring the Best Pattern Extension. In the prediction mode the user has a specific outcome in mind and wants to explore novel combinations of attribute values that are associated with this outcome. The user is aided by the best pattern extension, which orders the attributes in the Attribute Pane (Fig. 1B) according to their estimated impact on the results if added to the definition of the task in the Rule Pattern Pane.

Screencast 3: Data Preprocessing featuring Automatic Binning. Despite the availability of binning wildcards, it is more efficient to decrease the dimensionality of the attribute space in the preprocessing phase. Every manually specified binning is saved as a transformation scenario. In the automatic mode attributes in the dataset are compared with the saved scenarios using algorithms from the schema matching domain. If there is a sufficient match, the new attribute is binned according to a scenario defined earlier for a similar attribute.

Acknowledgements. The work described here was supported by grants IGA 26/2011, GACR 201/08/0802 of the Czech Science Foundation and the EU FP7 LinkedTV project. We thank Vojtěch Svátek for his feedback.

References

1. De Bie, T., Kontonasios, K.-N., Spyropoulou, E.: A framework for mining interesting pattern sets. In: Useful Patterns, UP 2010, pp. 27–35. ACM, New York (2010)
2. Goethals, B., Moens, S., Vreeken, J.: MIME: A Framework for Interactive Visual Pattern Mining. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 634–637. Springer, Heidelberg (2011)
3. Hájek, P., Holeňa, M., Rauch, J.: The GUHA method and its meaning for data mining. Journal of Computer and System Sciences 76, 34–48 (2010)
4. Kliegr, T., Hazucha, A., Marek, T.: Instant Feedback on Discovered Association Rules with PMML-Based Query-by-Example. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 257–262. Springer, Heidelberg (2011)
5. Kliegr, T., Svátek, V., Šimůnek, M., Ralbovský, M.: Semantic analytical reports: A framework for post-processing of data mining results. Journal of Intelligent Information Systems 37(3), 371–395 (2011)
6. Šimůnek, M., Tammisto, T.: Distributed Data-Mining in the LISp-Miner System Using Techila Grid. In: Zavoral, F., Yaghob, J., Pichappan, P., El-Qawasmeh, E. (eds.) NDT 2010. CCIS, vol. 87, pp. 15–20. Springer, Heidelberg (2010)

ASV Monitor: Creating Comparability of Machine Learning Methods for Content Analysis

Andreas Niekler[1], Patrick Jähnichen[2], and Gerhard Heyer[2]

[1] Faculty of Media, Leipzig University of Applied Sciences (HTWK)
[2] NLP Group, Department of Computer Science, University of Leipzig
aniekler@fbm.htwk-leipzig.de, {jaehnichen,heyer}@informatik.uni-leipzig.de

Abstract. In this demonstration paper we present an application to compare and evaluate machine learning methods used for natural language processing within a content analysis framework. Our aim is to provide an example set of possible machine learning results for different inputs to increase the acceptance of using machine learning in settings that originally rely on manual treatment. We will demonstrate the possibility to compare machine learning algorithms regarding the outcome of the implemented approaches. The application allows the user to evaluate the benefit of using machine learning algorithms for content analysis by a visual comparison of their results.

1 Introduction

Many emerging disciplines of the e-humanities like social sciences, media science, literature science or classical studies rely on content analysis theory. In those scientific fields a strong focus is laid on manual content analysis. Typically researchers in these fields are quite sceptical towards algorithmically driven content analysis approaches. With the presented application we provide a platform to motivate researchers in the humanities to compare and evaluate automatic methods for their own tasks using their own well known corpora. The freedom of importing corpora the contents of which are familiar will increase the acceptance of the automated methods. An almost confusing amount of different possible models for content analysis exist, facing researchers with the following problem: presenting the same data to different models naturally leads to outcomes that differ in its form or quality or both. This implies, that results from different models are not necessarily comparable and thus it is hard (if not impossible) to draw conclusions about the utility of using one model over the other. Another problem is the lack of visualisation of techniques of content analysis. Model implementations typically provide (if any) a rather rudimentary visualisation component, that is mostly comprised of ways to automatically generate HTML pages, showing the results in tables. A general way of visualising model outcome of different models for content analysis is clearly a goal to strive for. In

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 812–815, 2012.

section 2, we present the architecture of the ASV Monitor[1], a modular platform providing a wide range of machine learning techniques for content analysis that is easily extendable, preprocesses text data using common preprocessing steps, presents the preprocessed data to different algorithms and produces formalised model outcome. Passing the output to a visualisation component allows pictorial analysis of different models and thus provides a help for choosing the right one. The visualisation component is introduced in section 3 and an outlook on possible future extensions is given in section 4.

2 Architecture

Creating text corpora based content analysis applications requires the close interaction of algorithms and data structures. In our framework, data import is done by making use of a modular architecture which allows us to implement and add new importers for different kinds of sources. Within those, text is extracted and is then passed to sub-sequent modules doing noise reduction, i.e. stopword pruning, stemming and other standard preprocessing steps.

2.1 Algorithms

As content analysis mostly deals with unknown text sources, we implemented several unsupervised state-of-the-art machine learning algorithms to evaluate different aspects of text based content analysis. Analysts generally do not have any information about the nature of the text collection, so the problem to tackle is *not* a traditional classification problem, as in most cases we do not have training data, hence the focus on unsupervised algorithms. We concentrated on capturing the content analysis tasks defined by [1], which are: *summarisation*, *category and topic extraction*, *word context and word usage* and *meaning shift of topics and words over time*. We identified machine learning algorithms that can be adapted and aligned them to one of the above tasks. Again, the modularity of the system allows for new algorithms and visualisations to be included, thus enabling us to test and evaluate new ideas for using machine learning algorithms within content analysis tasks. Our current version comprises the following algorithms to address the different tasks:

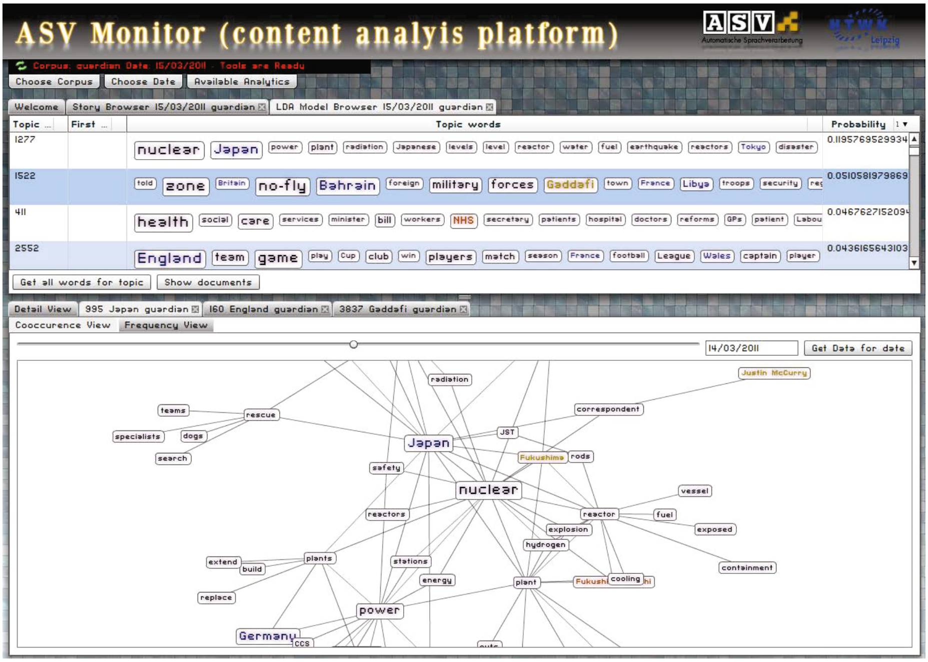
Topic Models: Latent Dirichlet Allocation[2] and Hierarchical Dirichlet Processes [3], *Tasks:* summarisation, category and topic extraction, meaning shift of topics and words over time
Topic Detection and Tracking [4], *Tasks:* summarisation, category and topic extraction, meaning shift of topics and words over time
Cooccurence analysis [5], *Tasks:* word context and word usage, meaning shift of topics and words over time.
Word burstiness [6], *Tasks:* word context and word usage.

[1] ASV stands for *Automatische SprachVerarbeitung*, a German translation of Natural Language Processing; the prototype can be accessed at `http://monitor.asv.informatik.uni-leipzig.de`, user credentials: `ecml_pkdd12/ecml`

2.2 Visualisation

The resulting visualisation is the core functionality of our framework since it provides the main access point to the results of a specific algorithm. Also, it enables us to critically discuss the decision for a particular algorithm with respect to a specific task.

The main screen of the application is a dashboard which allows the user to retrieve and interact with the results of the different algorithms as described above. We use Apache Flex[2] technology, because it provides a large number of ready to use components for the implementation of a data driven dashboard. Within the dashboard, the user is able to select a text corpus and review the results of different algorithms. To investigate time varying effects within a text collection, the framework provides access to corpora that span over a time period via a calendar-like interface. We split the main screen into a data grid centering on an informative overview of the algorithm's results (Fig. 1 top) and a detailed view showing the actual visualisations of word level aspects of the results (Fig. 1 bottom). There exist visualisations within the ASV Monitor for each of the implemented approaches. These include terms in different size according to their importance (as defined by the algorithm), different colors according to their NER labels.


ASV Monitor (content analyis platform)
ASV Automatische Sprachverarbeitung
HTWK Leipzig
Corpus: guardian Date: 15/03/2011 · Tools are Ready
Choose Corpus
Choose Date
Available Analytics
Welcome
Story Browser 15/03/2011 guardian
LDA Model Browser 15/03/2011 guardian

Topic ...	First ...	Topic words	Probability
1277		nuclear Japan power plant radiation Japanese levels level reactor water fuel earthquake reactors Tokyo disaster	0.1195769529934
1522		told zone Britain no-fly Bahrain foreign military forces Gaddafi town France Libya troops security reg	0.0510581979869
411		health social care services minister bill workers NHS secretary patients hospital doctors reforms GPs patient Labou	0.0467627152094
2552		England team game play Cup club win players match season France football League Wales captain player	0.0436165643103

Fig. 1. Screenshot of the main screen: Within the visualisation of summarisations and categories we use word clouds in different ways to visualize results

[2] http://incubator.apache.org/flex/

2.3 Comparison

In combination with text sources allocated by the user our aim is to provide means of comparing results of the analysis based on the algorithms and visualisations. This enables the user to compare and judge on the results against a previously known reference corpus. With a well known corpus, it is then easy to choose the best suited algorithm for a given task thus raising the acceptance of automated machine leaning methods. We focus on that task by providing the ability of presenting the different results side by side within the same browser window. We rely on tabbed browsing as a well known user interface to compare different visualisations to reach that goal.

3 Summarisation and Outlook

The ASV Monitor enables the user to compare the outcome of machine learning algorithms used to solve different content analysis tasks. It also provides an intuitive interface and the ability to use new text sources to evaluate algorithms on them. With this, it is easier to suggest appropriate algorithms and evaluate their weaknesses and strengths concerning certain tasks of content analysis. Our current version only uses algorithms that utilize the text content. Thus, an interesting extension could be the integration of methods that are able to analyze the behavior of time varying units like word frequencies or categories' document counts, as these are another important aspect that can be of use in content analysis e.g. for trend detection. Moreover, the whole domain of time series analysis could provide deeper insight for tasks dealing with diachronic data sources. Evaluation of those methods on real and recent text data provides better support to select the best approach for any given task.

References

1. Krippendorff, K.: Content Analysis: An Introduction To its Methodology, 2nd edn. Sage, Thousand Oaks (2004)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)
3. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Hierarchical Dirichlet Processes. Journal of the American Statistical Association (2006)
4. Allan, J.: Introduction to Topic Detection and Tracking. In: Allan, J., Croft, W.B. (eds.) Topic Detection and Tracking. The Information Retrieval Series. Springer US (2002)
5. Heyer, G.: Text Mining: Wissensrohstoff Text: Konzepte, Algorithmen, Ergebnisse. W3L-Verlag, Herdecke (2006)
6. Kleinberg, J.: Bursty and Hierarchical Structure in Streams. In: KDD 2002: Proceedings of the Eighth ACM SIGKDD (2002)

ClowdFlows: A Cloud Based Scientific Workflow Platform

Janez Kranjc[1,2], Vid Podpečan[1,2], and Nada Lavrač[1,2,3]

[1] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[3] University of Nova Gorica, Nova Gorica, Slovenia
{janez.kranjc,vid.podpecan,nada.lavrac}@ijs.si

Abstract. This paper presents an open cloud based platform for composition, execution, and sharing of interactive data mining workflows. It is based on the principles of service-oriented knowledge discovery, and features interactive scientific workflows. In contrast to comparable data mining platforms, our platform runs in all major Web browsers and platforms, including mobile devices. In terms of crowdsourcing, ClowdFlows provides researchers with an easy way to expose and share their work and results, as only an Internet connection and a Web browser are required to access the workflows from anywhere. Practitioners can use ClowdFlows to seamlessly integrate and join different implementations of algorithms, tools and Web services into a coherent workflow that can be executed in a cloud based application. ClowdFlows is also easily extensible during run-time by importing Web services and using them as new workflow components.

Keywords: cloud computing, data mining platform, service-oriented architecture, web application, web services, scientific workflows.

1 Introduction and Related Work

The paper presents ClowdFlows, an open cloud based platform for composition, execution and sharing of data mining workflows. It was designed to overcome deficiencies of existing comparable data mining platforms while retaining their useful features along with new features, not provided in comparable software. The presented platform is distinguished by these important features — a visual programming user interface that works in a Web browser, a service-oriented architecture that allows using third party services, a social aspect that allows sharing of scientific workflows, and a cloud-based execution of workflows. ClowdFlows is accessible online at http://clowdflows.org.

Tools for composition of workflows most often use the visual programming paradigm to implement the user interface. Notable applications that employ this approach include Weka [1], Orange [2], KNIME [3], and RapidMiner [4]. The most important common feature is the implementation of a *workflow canvas* where workflows can be constructed using simple drag, drop and connect

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 816–819, 2012.

operations on the available components. This feature makes the platforms suitable also for non-experts due to the representation of complex procedures as sequences of simple processing steps (workflow components).

In order to allow distributed processing, a service oriented architecture has been employed in platforms such as Orange4WS [5] and Taverna [6]. Utilization of Web services as processing components enables parallelization and remote execution. Service oriented architecture supports not only distributed processing but also distributed software development.

Sharing of workflows is a feature already implemented at the *myExperiment* website [6]. It allows users to publicly upload their workflows so that they are available to a wider audience and a link may be published in a research paper. However, the users that wish to view or execute these workflows are still required to install specific software in which the workflows were designed.

Remote workflow execution (on different machines than the one used for workflow construction) is also employed by RapidMiner using the RapidAnalytics server [4]. This allows the execution of workflows on more powerful machines and data sharing with other users, with the requirement that the client software is installed on the user's machine, which is a deficiency compared to our ClowdFlows solution.

With the described features in mind, we designed ClowdFlows, a platform which implements these features, but facilitated by enabling their access from a Web browser. The advantage of this approach is that no installation is required and that workflows may be run from any device with a modern Web browser, while being executed on the cloud. Apart from software and hardware independence, the implementation as a cloud based application takes all the processing load from the client's machine and moves it to the cloud where remote servers can run the experiments with or without user supervision.

2 The ClowdFlows Platform

ClowdFlows consists of the workflow editor (the graphical user interface) and the server side application which handles the execution of the workflows and hosts a number of publicly available workflows. The editor is implemented in HTML and JavaScript and runs in the client's browser. The server side is written in Python and uses the Django Web framework[1].

The workflow editor shown in Figure 1 consists of a *workflow canvas* and a *widget repository*. The widget repository is a list of all available workflow components which can be added to the workflow canvas. The repository includes a wide range of default widgets. Default widgets include Orange's implementation of classification algorithms, which were imported seamlessly as Orange is also implemented in Python. Weka's implementations of algorithms for classification and clustering, which we have wrapped as Web services, are also included in the widget repository by default. The widget repository may also be expanded by anyone at any time by importing Web services as workflow components by

[1] More information on Django can be found at `https://www.djangoproject.com/`

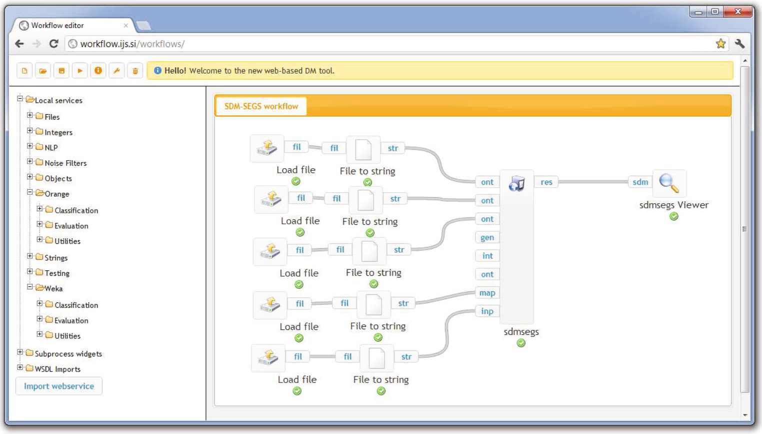

Hello! Welcome to the new web-based DM tool.

Fig. 1. A screenshot of the workflow editor with a semantic data mining workflow loaded [7]. This workflow can be accessed at the following address: http://clowdflows.org/workflow/104/.

entering a URL of a WSDL described Web service. The operations of the Web service are converted to widgets and their arguments and results are converted to inputs and outputs of these widgets. The repository also includes a set of process control widgets which allow the creation of meta-workflows (a workflow of workflows) and loops (meta-workflows that run multiple times), and a set of widgets for results visualization.

The server side consists of methods for the client side workflow editor to compose and execute workflows, and a relational database of workflows, widgets and data. The methods for manipulating workflows are accessed by the workflow editor using a series of asynchronous HTTP requests. Each request is handled by the server and executes widgets if necessary, saves the changes in the database and returns the results to the client. The server can handle multiple requests at a time and can simultaneously execute many workflows and widgets.

The data are stored on the server in the database. The platform is database independent, but MySQL is used in the public installation. The data can be passed as pointers or as the data itself, depending on the widget or Web service implementation.

A repository of public workflows which also serve as use cases for this demo is available in ClowdFlows and can be accessed athttp://clowdflows.org/existing-workflows/. Whenever the user opens a public workflow, a copy of that workflow appears in her private workflow repository in the workflow editor. The user can execute the workflow and view its results or expand it by adding or removing widgets. The user may again share her changes in the form of a new public workflow. Each public workflow can also be accessed through a unique address which is provided for the user to be shared through the workflow editor.

3 Conclusion and Further Work

We have developed an open and general cloud based platform for data mining, which employs service-oriented technologies, and is ready to be used in any data analysis scenario.

The social aspect of the platform provides a way for users to share their work easily as each workflow can be accessed through a simple address. This allows researchers to distribute their work and results with ease, since ClowdFlows provides cross-platform functionality. The platform is also suitable for non-experts and beginner data mining enthusiasts because of its intuitive and simple user interface.

We are currently working on adding support for mining continuous data streams from the Web (e.g. RSS feeds). We will also continue to add new widgets for specialized machine learning and data mining tasks, focusing on text mining.

Acknowledgments. This work was supported by the FP7 European Commission projects “Machine understanding for interactive storytelling” (MUSE, grant agreement no: 296703) and “Large scale information extraction and integration infrastructure for supporting financial decision making” (FIRST, grant agreement 257928).

References

1. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann, Amsterdam (2011)
2. Demšar, J., Zupan, B., Leban, G., Curk, T.: Orange: From Experimental Machine Learning to Interactive Data Mining. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 537–539. Springer, Heidelberg (2004)
3. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R. (eds.) GfKl. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 319–326. Springer (2007)
4. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: Ungar, L., Craven, M., Gunopulos, D., Eliassi-Rad, T. (eds.) KDD 2006: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 935–940. ACM, New York (2006)
5. Podpečan, V., Zemenova, M., Lavrač, N.: Orange4ws environment for service-oriented data mining. The Computer Journal 55(1), 89–98 (2012)
6. Hull, D., Wolstencroft, K., Stevens, R., Goble, C.A., Pocock, M.R., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. Nucleic Acids Research 34(web-server-issue), 729–732 (2006)
7. Lavrač, N., Vavpetič, A., Soldatova, L.N., Trajkovski, I., Novak, P.K.: Using ontologies in semantic data mining with segs and g-segs. Discovery Science, 165–178 (2011)

Extracting Trajectories through an Efficient and Unifying Spatio-temporal Pattern Mining System

Phan Nhat Hai[1,2], Dino Ienco[1,2],
Pascal Poncelet[1,2], and Maguelonne Teisseire[1,2]

[1] IRSTEA Montpellier, UMR TETIS - 34093 Montpellier, France
{nhat-hai.phan,dino.ienco,maguelonne.teisseire}@teledetection.fr
[2] LIRMM CNRS Montpellier - 34090 Montpellier, France
pascal.poncelet@lirmm.fr

Abstract. Recent improvements in positioning technology has led to a much wider availability of massive moving object data. A crucial task is to find the moving objects that travel together. Usually, these object sets are called spatio-temporal patterns. Analyzing such data has been applied in many real world applications, e.g., in ecological study, vehicle control, mobile communication management, etc. However, few tools are available for flexible and scalable analysis of massive scale moving objects. Additionally, there is no framework devoted to efficiently manage multiple kinds of patterns at the same time. Motivated by this issue, we propose a framework, named GeT_Move, which is designed to extract and manage different kinds of spatio-temporal patterns concurrently. A user-friendly interface is provided to facilitate interactive exploration of mining results. Since GeT_Move is tested on many kinds of real data sets, it will benefit users to carry out versatile analysis on these kinds of data by exhibiting different kinds of patterns efficiently.

1 Introduction

Nowadays, many electronic devices are used for real world applications. Telemetry attached on wildlife, GPS installed in cars, sensor networks, and mobile phones have enabled the tracking of almost any kind of moving object data and has led to an increasingly large amount of data. Therefore, analysis on such data to find interesting patterns, called spatio-temporal patterns, is attracting increasing attention for applications such as movement pattern analysis, animal behavior study, route planning and vehicle control.

Despite the growing demands for diverse applications, there have been few scalable tools for mining massive and sophisticated moving object data. Even if some tools are available for extracting patterns (e.g. [4]), they mainly focus on specific kinds of patterns at a time. Obviously, when considering a dataset, it is quite difficult, for the decision maker, to know in advance which kinds of patterns are embedded in the data. To cope with this issue, we propose the GeT_Move system to reveal, automatically and in a very efficient way, collective movement

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 820–823, 2012.

patterns like convoys [1], group patterns [5], closed swarms [3], moving clusters [2] and also periodic patterns [7]. Starting from the results of GeT_Move, the user can then visualize, browse and compare the different extracted patterns through a user friendly interface.

2 Spatio-temporal Patterns

For clarity sake, we briefly remind the spatio-temporal pattern definitions. Informally, a *swarm* is a group of moving objects O containing at least min_o individuals which are closed each other for at least min_t timestamps T. To avoid redundant swarms, Zhenhui Li et al. [3] propose the notion of *closed swarm* for grouping together both objects and time. A swarm (O, T) is a closed swarm if it cannot be enlarged in terms of timestamps T and objects O. Another pattern is convoy which is also a group of objects such that these objects are closed each other during at least min_t time points. The main difference between convoy and swarm (or closed swarm) is that convoy lifetimes must be consecutive. Furthermore, moving clusters can be seen as special cases of convoys with the additional condition that they need to share some objects between two consecutive timestamps [6]. We can consider that the main difference between convoys and swarms is about the consecutiveness and non-consecutiveness of clusters during a time interval. In [5], Hwang et al. propose a general pattern, called a *group pattern*, which essentially is a combination of both convoys and closed swarms. Basically, group pattern is a set of disjointed convoys which are generated by the same group of objects in different time intervals. By considering a convoy as a time point, a group pattern can be seen as a closed swarm of disjointed convoys.

3 The GeT_Move System Architecture

The GeT_Move general architecture, described in Figure 1, has three main layers: (i) collection and cleaning, (ii) mining, and (iii) visualization. The bottom layer is responsible for collecting and preprocessing moving objects. During this step, some cleaning and interpolations techniques are used to integrate and clean the raw data as well as to interpolate missing points. The interpolation techniques used are similar to the ones provided by most of spatio-temporal pattern mining algorithms.

GeT_Move uses a new mining algorithm able to exploit the similar characteristics among different kinds of patterns. This new mining method combines both clustering and pattern mining to extract the final results. So, by applying it to the preprocessed data, GeT_Move can automatically extract different kinds of patterns such as convoys, closed swarms, group patterns, moving clusters and periodic patterns.

The top layer is devoted to the visualization. GeT_Move provides a platform for users to flexibly tune parameters and supports visualization of the results in different formats. The output can be written in Google Map[1] and Google Earth[2] formats to help users better explore the results. Furthermore, the system enables users to explore, plot and navigate over the different patterns in order to compare the differences among the various moving objects behaviors.

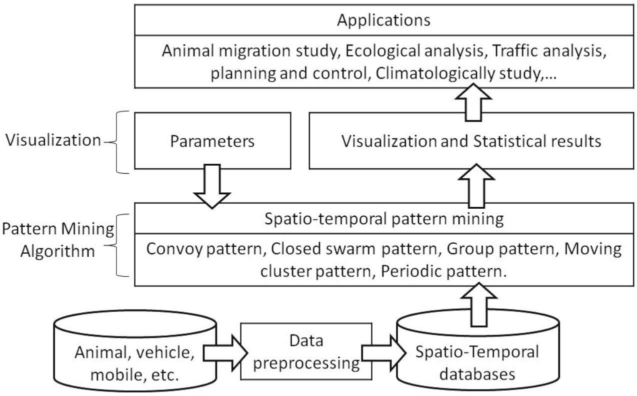
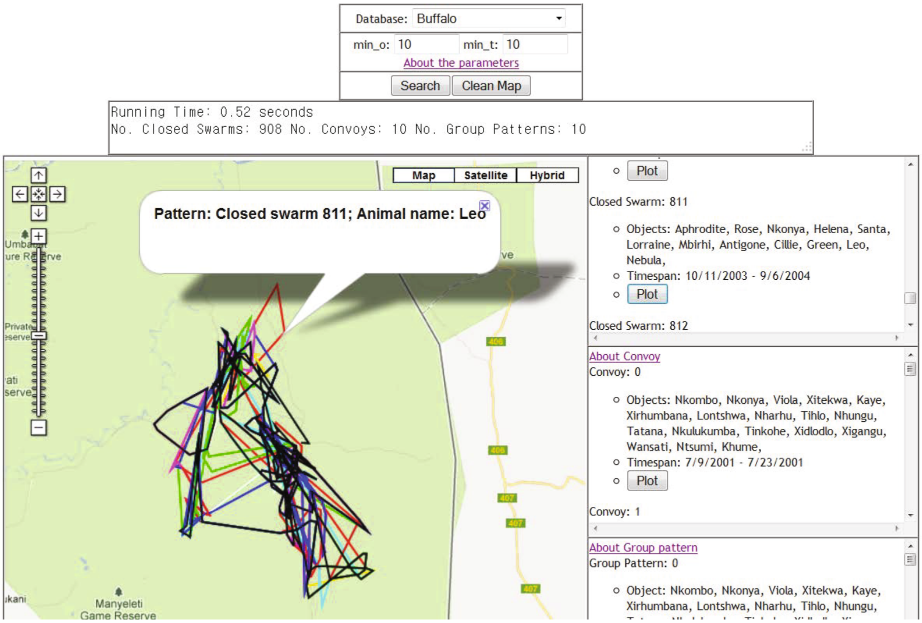

Applications
Animal migration study, Ecological analysis, Traffic analysis, planning and control, Climatologically study,...
Visualization
Parameters
Visualization and Statistical results
Pattern Mining Algorithm
Spatio-temporal pattern mining
Convoy pattern, Closed swarm pattern, Group pattern, Moving cluster pattern, Periodic pattern.
Animal, vehicle, mobile, etc.
Data preprocessing
Spatio-Temporal databases

Fig. 1. The GeT_Move System Architecture


Database: Buffalo
min_o: 10 min_t: 10
About the parameters
Search Clean Map

```
Running Time: 0.52 seconds
No. Closed Swarms: 908 No. Convoys: 10 No. Group Patterns: 10
```

Pattern: Closed swarm 811; Animal name: Leo

- Plot

Closed Swarm: 811

- Objects: Aphrodite, Rose, Nkonya, Helena, Santa, Lorraine, Mbirhi, Antigone, Cillie, Green, Leo, Nebula,
- Timespan: 10/11/2003 - 9/6/2004
- Plot

Closed Swarm: 812

About Convoy
Convoy: 0

- Objects: Nkombo, Nkonya, Viola, Xitekwa, Kaye, Xirhumbana, Lontshwa, Nharhu, Tihlo, Nhungu, Tatana, Nkulukumba, Tinkohe, Xidlodlo, Xigangu, Wansati, Ntsumi, Khume,
- Timespan: 7/9/2001 - 7/23/2001
- Plot

Convoy: 1

About Group pattern
Group Pattern: 0

- Object: Nkombo, Nkonya, Viola, Xitekwa, Kaye, Xirhumbana, Lontshwa, Nharhu, Tihlo, Nhungu,

Fig. 2. The Graphical Visualization Interface

[1] http://code.google.com/apis/maps/

[2] http://earth.google.com/

Figure 2 shows a screenshot of GeT_Move[3]. It allows to choose a dataset and set the values of the parameter min_o and min_t. In this example we have chosen the *Buffalo* dataset from the Movebank[4] project with $min_o = 10$ and $min_t = 10$ (combo box in the top of the figure). This dataset concerns the raw trajectories of 165 Buffaloes gathered from year 2000 to year 2006. In the second combox box in the figure, we can observe that, with these parameters, there are 908 closed swarms, 10 convoys and 10 group patterns. In this example, we observe that the execution time for extracting all these patterns is 0.52 seconds. From the combo box on right, the user can thus select a specific pattern and plot it on the main window. For instance, in our example, the user has decided to plot the 811^{th} Closed Swarm. Finally the user can have more information such as pattern identifier, animal name and timespan when clicking on plotted trajectories as illustrated in the main window.

4 Conclusion

In this paper, we propose a system, GeT_Move, which is designed to automatically and efficiently extract different kinds of spatio-temporal patterns at the same time. Starting from these results, the user can browse, navigate and compare different patterns in an easy way. The comparative analysis allows the user to understand and discover which results can fit better her researches.

References

1. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S., Shen, H.T.: Discovery of Convoys in Trajectory Databases. PVLDB 1(1), 1068–1080 (2008)
2. Kalnis, P., Mamoulis, N., Bakiras, S.: On Discovering Moving Clusters in Spatio-temporal Data. In: Medeiros, C.B., Egenhofer, M., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 364–381. Springer, Heidelberg (2005)
3. Li, Z., Ding, B., Han, J., Kays, R.: Swarm: Mining Relaxed Temporal Moving Object Clusters. In: VLDB 2010, Singapore, pp. 723–734 (2010)
4. Li, Z., Ji, M., Lee, J.-G., Tang, L., Yu, Y., Han, J., Kays, R.: Movemine: Mining moving object databases. In: SIGMOD 2010, Indianapolis, Indiana, pp. 1203–1206 (2010)
5. Wang, Y., Lim, E.-P., Hwang, S.-Y.: Efficient Mining of Group Patterns from User Movement Data. In: DKE, pp. 240–282 (2006)
6. Han, J., Li, Z., Tang, L.A.: Mining Moving Object, Trajectory and Traffic Data. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5982, pp. 485–486. Springer, Heidelberg (2010)
7. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, Indexing, and Querying Historical Spatiotemporal Data. In: SIGKDD 2004, pp. 236–245 (2004)

[3] http://www.lirmm.fr/~phan/index.jsp

[4] http://www.movebank.org/

Knowledge Discovery through Symbolic Regression with HeuristicLab

Gabriel Kronberger, Stefan Wagner, Michael Kommenda, Andreas Beham,
Andreas Scheibenpflug, and Michael Affenzeller

University of Applied Sciences Upper Austria, Campus Hagenberg
School for Informatics, Communication and Media
{gkronber,swagner,mkommend,abeham,ascheibe,maffenze}@heuristiclab.com
http://heal.heuristiclab.com/

Abstract. This contribution describes how symbolic regression can be used for knowledge discovery with the open-source software HeuristicLab. HeuristicLab includes a large set of algorithms and problems for combinatorial optimization and for regression and classification, including symbolic regression with genetic programming. It provides a rich GUI to analyze and compare algorithms and identified models. This contribution mainly focuses on specific aspects of symbolic regression that are unique to HeuristicLab, in particular, the identification of relevant variables and model simplification.

1 Introduction

HeuristicLab[1] is an open-source software system for heuristic optimization that features several metaheuristic optimization algorithms as well as several optimization problems. It is implemented in C# and based on the Microsoft .NET Framework and provides a rich graphical user interface for solving optimization problems and for experiment analysis. HeuristicLab is used in a number of fundamental and practical research projects [1], as well as in lectures on data mining and machine learning. In 2009 the project achieved the second place of the Microsoft Innovation Award in Austria and is released under the GNU General Public License. Some of the general features of HeuristicLab 3.3 are:

- comfortable and feature rich graphical user interface
- experiment designer to create and execute a large number of test runs
- graphical analysis and comparison of parameters and results
- graphical algorithm designer to create or modify algorithms
- plug-in based architecture which enables an easy integration of new algorithms and problems

Based on these general features HeuristicLab provides several well-known algorithms for classification and regression (e.g. linear regression, random forest,

[1] http://dev.heuristiclab.com
http://dev.heuristiclab.com/AdditionalMaterial/ECML-PKDD

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 824–827, 2012.

SVM, ...), which can be easily configured and applied to a given data set directly in the GUI. The experiment designer simplifies algorithm configuration and makes it very easy to find the best settings for a given problem.

Additionally to standard methods for regression HeuristicLab also provides an extensive implementation of symbolic regression based on genetic programming [2],[3]. In this contribution we show the core principle of symbolic regression and discuss how this method can be used for practical data mining applications.

2 User Groups and Related Work

HeuristicLab users can be categorized into three relevant groups: practitioners, experts, and students. Practitioners are trying to solve real-world problems with classical and advanced algorithms. Experts include researchers and graduate students who are developing new advanced algorithms. Students can learn about standard algorithms and can try different algorithms and parameter settings to various benchmark algorithms. The design and architecture of HeuristicLab is specifically tuned to these three user groups and we put a strong emphasis on the ease of use of the software.

Several software systems for symbolic regression or more generally for metaheuristic optimization have been developed in the recent years. *Formulize*[2] (or Eureqa-II) is notable as it also provides a user friendly GUI, uses powerful state-of-the-art algorithms and also provides a simple way to execute runs in the Cloud. However, it is much more specialized than HeuristicLab and is mainly designed for practitioners. *ECJ*[3] is an evolutionary computation system written in Java. It is well established as an implementation and experimentation platform in the EC community and also includes a framework for symbolic regression. A drawback ECJ is its limited GUI which makes it difficult to for example analyze the prediction accuracy of the discovered models.

3 Case Study: Tower Data

In this contribution we demonstrate the unique features of HeuristicLab and how it can be used for knowledge discovery in a real world application, in particular, for finding relevant driving factors in a chemical process and for the identification of white-box regression models for the process. The analysis is based on the *tower data set* which is kindly provided by Dr. Arthur Kordon from Dow Chemical [4]. In the demonstration we show the exact process to achieve the results, that can only be briefly described in the following two sections because of page constraints.

3.1 Identification of Non-linear Models

The following equation shows a non-linear model for the tower data set as identified by symbolic regression in HeuristicLab. The algorithm discovered the model

[2] http://www.nutonian.com/eureqa-ii/

[3] http://cs.gmu.edu/~eclab/projects/ecj/

structure and parameters automatically through an evolutionary process by assembling the basic building blocks: random constants, input variables, arithmetic operators, and the logarithm and exponential functions.

$$\frac{c_0 \cdot \mathrm{x}23 \cdot \left(((c_1 \cdot \mathrm{x}5 - c_2 \cdot \mathrm{x}6) - c_3 \cdot \mathrm{x}14) + \left(\frac{(c_4 \cdot \mathrm{x}6 - c_5 \cdot \mathrm{x}14)}{(c_6 \cdot \mathrm{x}1 - c_7 \cdot \mathrm{x}4)} - (c_8 - c_9 \cdot \mathrm{x}4) \right) \right)}{c_{16} \cdot \mathrm{x}1} \cdot c_{17}$$
$$+ \frac{(c_{10} \cdot \mathrm{x}5 - c_{11} \cdot \mathrm{x}6) - (c_{12} \cdot \mathrm{x}4 - (\log(c_{13} \cdot \mathrm{x}4) - (c_{14} \cdot \mathrm{x}5 - c_{15} \cdot \mathrm{x}24)))}{c_{16} \cdot \mathrm{x}1} \cdot c_{17} + c_{18}$$

3.2 Identification of Relevant Variables

Frequently it is not necessary to learn a full model of the functional relationship but instead only find a set of relevant variables for the process. This can be achieved easily with HeuristicLab through analysis of relative variable frequencies in the population of models. Figure 1 shows a variable frequency chart that clearly shows the six most relevant variables. Notably, the relevance of variables is determined based on non-linear models. So, non-linear influence factors and pair-wise interacting factors can be identified as well.

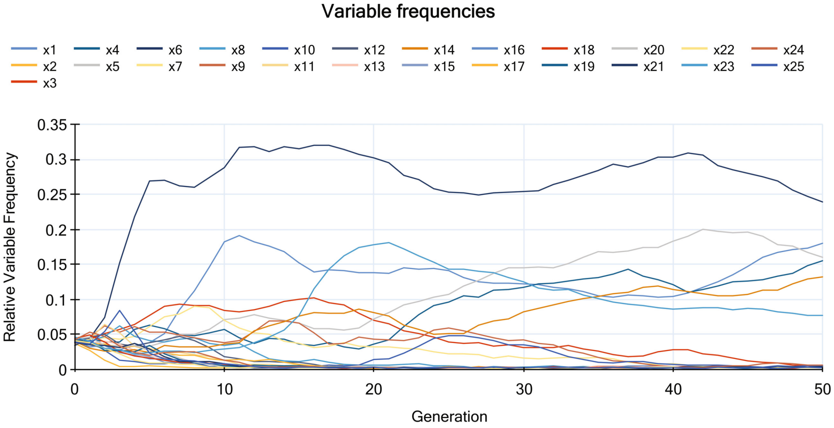
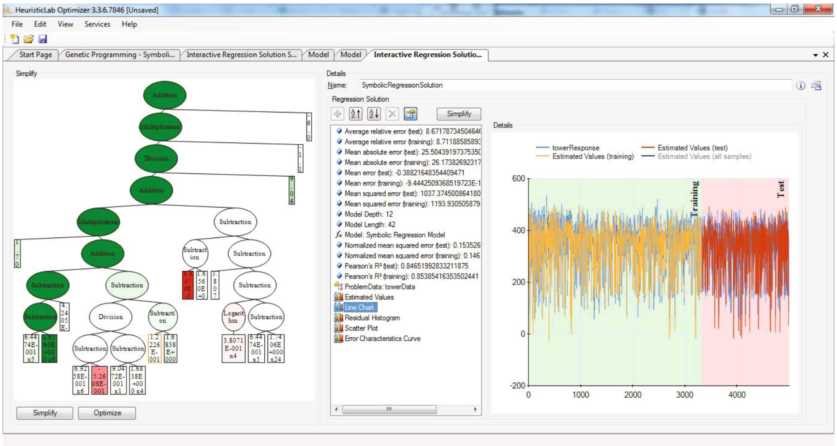


Fig. 1. Evolution of referenced input variables over a symbolic regression run

3.3 Simplification of Models with Visual Hints

A unique feature of HeuristicLab are visual hints for model simplification. By means of visual hints it is very easy to manually prune a complex model as shown above to find a good balance between complexity and accuracy. Figure 2 shows the GUI for model simplification. Green model fragments have a strong impact on the model output while white fragments can be pruned with minimal



Name: SymbolicRegressionSolution

- Average relative error (test): 8.67178734504646
- Average relative error (training): 8.71188585893
- Mean absolute error (test): 25.50439197375350
- Mean absolute error (training): 26.17382692317
- Mean error (test): -0.38821648354409471
- Mean error (training): -9.4442509368519723E-1
- Mean squared error (test): 1037.374500864180
- Mean squared error (training): 1193.930505879
- Model Depth: 12
- Model Length: 42
- Model: Symbolic Regression Model
- Normalized mean squared error (test): 0.153526
- Normalized mean squared error (training): 0.146
- Pearson's R² (test): 0.84651992833211875
- Pearson's R² (training): 0.85385416353502441
- ProblemData: towerData
- Estimated Values
- Line Chart
- Residual Histogram
- Scatter Plot
- Error Characteristics Curve

Fig. 2. Visual hints guide the user in manual simplification of non-linear models. Charts and accuracy metrics are updated in real-time to support the user.

losses in accuracy, in contrast red fragments increase the error of the model and should be removed. The GUI for model simplification immediately recalculates all error metrics and updates charts dynamically whenever a part of the model is changed by the user.

References

1. Affenzeller, M., Winkler, S., Wagner, S., Beham, A.: Genetic Algorithms and Genetic Programming - Modern Concepts and Practical Applications. Numerical Insights. CRC Press (2009)
2. Kronberger, G.: Symbolic Regression for Knowledge Discovery - Bloat, Overfitting, and Variable Interaction Networks. Trauner Verlag, Linz (2011)
3. Veeramachaneni, K., Vladislavleva, E., O'Reilly, U.-M.: Knowledge mining sensory evaluation data: genetic programming, statistical techniques, and swarm optimization. Genetic Programming and Evolvable Machines 13(1), 103–133 (2012)
4. Vladislavleva, E.J., Smits, G.F., den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation 13(2), 333–349 (2009)

OutRules: A Framework for Outlier Descriptions in Multiple Context Spaces

Emmanuel Müller, Fabian Keller, Sebastian Blanc, and Klemens Böhm

Karlsruhe Institute of Technology (KIT), Germany
{emmanuel.mueller,fabian.keller,klemens.boehm}@kit.edu,
sebastian.blanc@student.kit.edu

Abstract. Analyzing exceptional objects is an important mining task. It includes the identification of outliers but also the description of outlier properties in contrast to regular objects. However, existing *detection* approaches miss to provide important *descriptions* that allow human understanding of outlier reasons. In this work we present *OutRules*, a framework for outlier descriptions that enable an easy understanding of multiple outlier reasons in different contexts. We introduce outlier rules as a novel outlier description model. A rule illustrates the deviation of an outlier in contrast to its context that is considered to be normal. Our framework highlights the practical use of outlier rules and provides the basis for future development of outlier description models.

1 Open Challenges in Outlier Description

Outlier mining focuses on unexpected, rare, and suspicious objects in large data volumes [4]. Examples of outliers could be fraudulent activities in financial transaction records or unexpected patient behavior in health databases. Outlier mining has two aspects: (1) *identification* and (2) *description* of outliers. A multitude of approaches has been established for the former task (e.g., LOF [3] and more recent algorithms). They all focus on the quantification of outlierness, i.e., how strongly an object deviates from the residual data. Following this development of outlier detection algorithms there have been extensions of toolkits like *WEKA*, *RapidMiner*, and *R*, and stand-alone toolkits such as *ELKI* have been proposed. In all cases, only outlier detection algorithms have been implemented, and raw outlier results are visualized in different ways.

In contrast to this focus on the identification of outliers, approaches supporting outlier descriptions have been developed recently [6,1,2,7,9,10,5]. They aim at the description of the object's deviation, e.g. by selecting the deviating attributes for each individual outlier. These techniques assist humans in verifying the outlier characteristics. Without such outlier descriptions, humans are overwhelmed by outlier results that cannot be verified manually due to large and high dimensional databases. Humans might miss outlier reasons, especially if outliers are deviating w.r.t. multiple contexts. Therefore, humans depend on appropriate descriptions. This situation enforces the development of novel outlier description algorithms and their comparison in a unified framework.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 828–832, 2012.

2 The *OutRules* Framework

With *OutRules*[1] we extend our outlier mining framework [8], which is based on the popular *WEKA* toolkit. *OutRules* extracts both regular and deviating attribute sets for each outlier and presents them as so-called outlier rules. We utilize the cognitive abilities of humans by allowing a comparison of the outlier object vs. its regular context. This comparison enables an easy understanding of the individual outlier characteristics. In a health-care example with attributes *age*, *height*, and *weight* (cf. Fig. 1), a description for the marked outlier could be "the outlier deviates w.r.t. (1) *height* and *weight* and (2) *height* and *age*". However, this first description provides the deviating attribute combinations only. In addition, we present groups of clustered objects (e.g., in attributes *weight* and *age*) as the regular contexts of the outlier. Overall, we present multiple contexts as regular neighborhoods from which the outlier is deviating. Reasoning is then enabled by manual comparison and exploration of these context spaces.

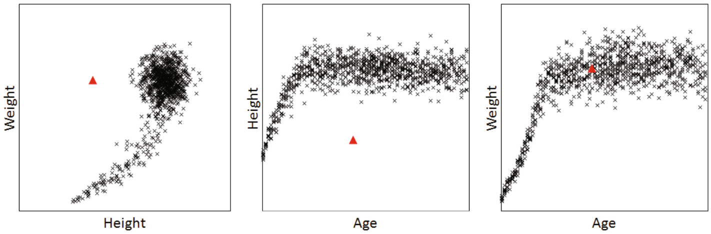


Fig. 1. Example of an outlier deviating w.r.t. multiple contexts

Outlier Rules as Basis for Outlier Descriptions

Our description model is based on the intuitive observation that each outlier deviates from other objects that are considered to be normal. Outlier rules accordingly represent these antagonistic properties of regularity on the one side and irregularity on the other side. As depicted in our example, there are multiple attribute combinations in which the object is an outlier, and there are multiple contexts in which it is regular. Several recent publications have observed this multiplicity of context spaces [1,7,9,10,5]. *OutRules* is the first framework that exploits these multiple context spaces for outlier rules. It illustrates the similarity among clustered objects and the deviation of the individual outlier. Therefore, it provides information about multiple contexts and highlights the differences to its local neighborhoods in these context spaces.

We consider each outlier individually and compute multiple outlier rules for each object. Each outlier rule is a set of attributes that show highly clustered objects on the one side, and on the other side, an extended set of attributes in which one of these objects is highly deviating. For instance in our previous example the outlier occurs under the attributes *age* and *height*. A first rule

[1] Project website: `http://www.ipd.kit.edu/~muellere/OutRules/`

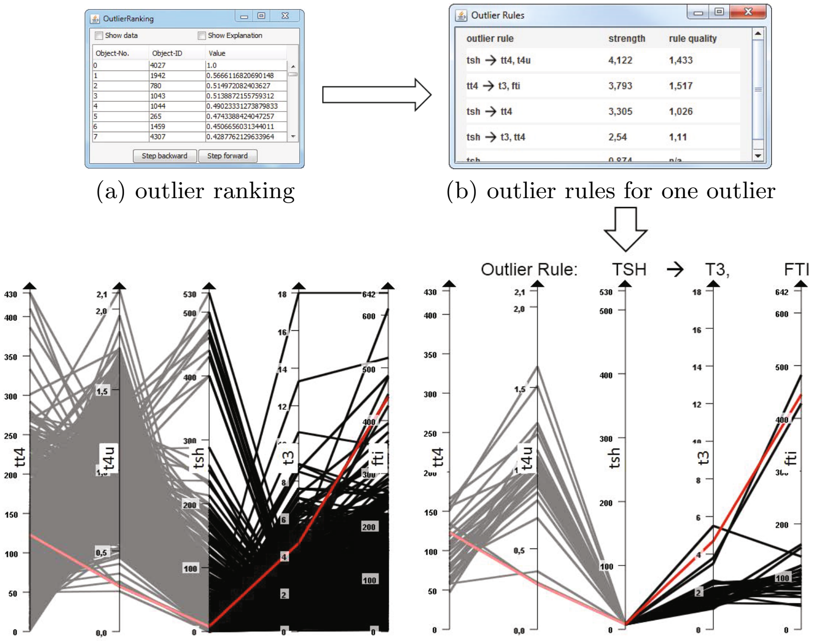
could be "The age is normal but the person is significantly too short". In this case the description might lead to the casual explanation that the represented person suffers from impaired growth. This outlier rule can be represented as $\{age\} \Rightarrow \{height\}$. Formally, an outlier rule is defined as follows:

Definition 1. ***Outlier Rule*** $A \Rightarrow B$

For an object o, the rule $A \Rightarrow B$ describes the *cluster membership* of o in attribute set $A \subseteq Attributes$ and the *deviating behavior* in $A \cup B \subseteq Attributes$.

The notion of *clustered* and *deviating* behavior can be instantiated by the underlying outlier score, e.g. by the notion of density in case of LOF [3].

We call A the *context* of o in which it shows regular behavior. As depicted in our example, there might be multiple reasons for an outlier deviation. Hence, our algorithm has to detect multiple contexts in which o is clustered. As the actual reason for an outlier is highly application-dependent, it is hard to make a binary decision of relevant and irrelevant rules. Therefore we output a ranking of all extracted rules. We rate each rule based on the data distribution in A and $A \cup B$. Based on the fact that an outlier rule represents the degree of regularity to other objects in the left hand side A and the degree of outlierness in the right hand side $A \cup B$, it is clear that the criteria have to quantify these two aspects. In our framework we have implemented criteria such as the *strength* of the outlier rule. It is defined as an instantiation of the well-established density-based outlier



OutlierRanking

☐ Show data ☐ Show Explanation

Object-No.	Object-ID	Value
0	4027	1.0
1	1942	0.5666116820690148
2	780	0.514972082403627
3	1043	0.5138872155759312
4	1044	0.49023331273879833
5	265	0.4743388424047257
6	1459	0.4506656031344011
7	4307	0.4287762129633964

(a) outlier ranking

Outlier Rules

outlier rule	strength	rule quality
tsh → tt4, t4u	4,122	1,433
tt4 → t3, fti	3,793	1,517
tsh → tt4	3,305	1,026
tsh → t3, tt4	2,54	1,11
tsh	0,874	n/a

(b) outlier rules for one outlier

(c) parallel coordinates plots; left: no context; right: neighborhood in TSH

Fig. 2. One exemplary outlier from the Thyroid data set [UCI ML repository]

scoring [3]. Please note that the framework is open for any instantiation of quality criteria, e.g. for outlier rules in a specific application scenario.

Visualization of Outlier Rules

The visualization of outlier rules consists of three components. An overview of outliers is presented in the outlier ranking component (cf. Fig. 2(a)). Individual outliers can be chosen from this ranking for further exploration. The second component is a list of outlier rules sorted by the strength or other quality measures (cf. Fig. 2(b)). The last component is the visualization of individual outlier rules; each outlier rule can be explored in more detail by looking at the underlying data distribution. For example, we have implemented scatter plots, distribution statistics, density-distributions in individual attributes, and more enhanced visual representations such as well-established parallel coordinate plots (cf. Fig. 2(c)). As illustrated by the parallel coordinate plots for a real world example, *Thyroid Disease* from the UCI ML repository, the properties of the outlier rule and the nature of the outlier become clearer by the comparison with similar objects. If we consider all objects in the database in the left plot, we observe that the outlier is quite regular for all attributes from a global point of view. However, if one restricts the visualization to its local neighborhood in attribute TSH in the right plot there is a clear cluster containing the outlier, while attributes $T3$ and FTI show high deviation for the outlier from the local neighborhood. The clustering in TSH and the deviation in $\{T3, FTI\}$ indicate the correctness of this rule in a real world example.

Acknowledgments. This work has been supported by YIG "Outlier Mining in Heterogeneous Data Spaces" and RSA "Descriptive Outlier Mining" funded by KIT as part of the German Excellence Initiative; and by the German Research Foundation (DFG) within GRK 1194.

References

1. Aggarwal, C.C., Yu, P.S.: Outlier detection for high dimensional data. In: SIGMOD, pp. 37–46 (2001)
2. Angiulli, F., Fassetti, F., Palopoli, L.: Detecting outlying properties of exceptional objects. ACM Trans. Database Syst. 34(1), 1–62 (2009)
3. Breunig, M., Kriegel, H.-P., Ng, R., Sander, J.: LOF: Identifying density-based local outliers. In: SIGMOD, pp. 93–104 (2000)
4. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. 41(3) (2009)
5. Keller, F., Müller, E., Böhm, K.: HiCS: High contrast subspaces for density-based outlier ranking. In: ICDE, pp. 1037–1048 (2012)
6. Knorr, E.M., Ng, R.T.: Finding intensional knowledge of distance-based outliers. In: VLDB, pp. 211–222 (1999)
7. Kriegel, H.-P., Kröger, P., Schubert, E., Zimek, A.: Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 831–838. Springer, Heidelberg (2009)

8. Müller, E., Schiffer, M., Gerwert, P., Hannen, M., Jansen, T., Seidl, T.: *SOREX*: Subspace Outlier Ranking Exploration Toolkit. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part III. LNCS, vol. 6323, pp. 607–610. Springer, Heidelberg (2010)
9. Müller, E., Schiffer, M., Seidl, T.: Statistical selection of relevant subspace projections for outlier ranking. In: ICDE, pp. 434–445 (2011)
10. Smets, K., Vreeken, J.: The odd one out: Identifying and characterising anomalies. In: SDM, pp. 804–815 (2011)

Scientific Workflow Management with ADAMS

Peter Reutemann[1] and Joaquin Vanschoren[2]

[1] University of Waikato, Hamilton, NZ
fracpete@waikato.ac.nz
[2] Leiden University, Leiden, NL
joaquin@liacs.nl

Abstract. We demonstrate the Advanced Data mining And Machine learning System (ADAMS), a novel workflow engine designed for rapid prototyping and maintenance of complex knowledge workflows. ADAMS does not require the user to manually connect inputs to outputs on a large canvas. It uses a compact workflow representation, *control operators*, and a simple interface between operators, allowing them to be auto-connected. It contains an extensive library of operators for various types of analysis, and a convenient plug-in architecture to easily add new ones.

Keywords: scientific workflows, machine learning, data mining.

1 Introduction

Many of today's data mining platforms offer *workflow engines* allowing the user to design and run knowledge workflows, from cleaning raw data to building models and making predictions. Most of these systems, such as Kepler [1], RapidMiner [2] and KNIME [3], represent these dependencies in a directed graph.[1] Many of them take a "canvas"-based approach, in which the user places operators on a large canvas and then connects the various inputs and outputs manually, thus introducing each dependency as a line on the canvas.

Though this is a very intuitive approach that greatly appeals to many end users, it is also a very time consuming one. When inserting additional operators, one has to move and rearrange the entire workflow to keep the design tidy. If an operator is replaced, all connections have to be redrawn. Moreover, scientific workflows often grow very complex, including hundreds of independent steps [5]. On a canvas, this leads to very large and complex graphs with many interconnections. This means that oversight is easily lost, even with useful features such as zooming, hierarchical workflows or meta-operators with internal workflows.

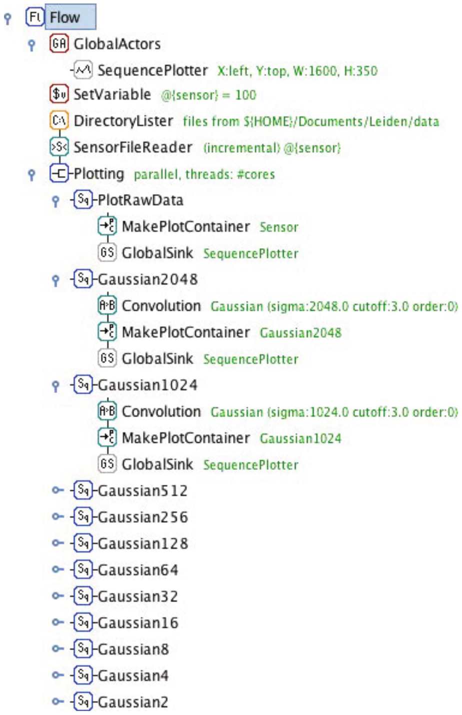
In this paper, we present ADAMS (Advanced Data mining And Machine learning System), a novel workflow engine specifically designed for rapid prototyping of complex scientific workflows, taking away the need to manually lay out and connect operators on a canvas. It presents the workflow in a compact tree structure in which operators can quickly be dragged in or pulled out, and

[1] For an in-depth overview and comparison of scientific workflow systems, see [4,5].

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 833–837, 2012.

auto-connected to the surrounding operators. It includes an extensive library of operators, including a range of *control operators* to create and direct sub-flows. Moreover, new operators can be added very easily, either by dropping them in a folder, or writing them on-the-fly, without compilation, in scripts.

2 Workflow Representation



```
Flow
  GlobalActors
    SequencePlotter  X:left, Y:top, W:1600, H:350
  SetVariable  @{sensor} = 100
  DirectoryLister  files from ${HOME}/Documents/Leiden/data
  SensorFileReader  (incremental) @{sensor}
  Plotting  parallel, threads: #cores
    PlotRawData
      MakePlotContainer  Sensor
      GlobalSink  SequencePlotter
    Gaussian2048
      Convolution  Gaussian (sigma:2048.0 cutoff:3.0 order:0)
      MakePlotContainer  Gaussian2048
      GlobalSink  SequencePlotter
    Gaussian1024
      Convolution  Gaussian (sigma:1024.0 cutoff:3.0 order:0)
      MakePlotContainer  Gaussian1024
      GlobalSink  SequencePlotter
    Gaussian512
    Gaussian256
    Gaussian128
    Gaussian64
    Gaussian32
    Gaussian16
    Gaussian8
    Gaussian4
    Gaussian2
```

Fig. 1. Data visualization flow

Figure 1 illustrates an ADAMS workflow. It reads files from a directory with sensor data and plots the raw and convoluted data (scale-space composition). Operators (called 'actors') are dragged from a library into the flow, and will automatically 'snap' into the tree structure depending on where they are dropped. Actors are shown as nodes with a name and a list of parameter settings (options). They are color-coded based on whether they are a *source* (only output, e.g. `DirectoryLister` (C:\)), *transformer* (input and output, e.g. `Convolution` (A>B)), *sink* (only input, e.g. `SequencePlotter` (∿)) or *standalone* (no in/output, e.g. `GlobalActors` (GA)). Branches can be collapsed, and clicking actors opens a settings dialog. Some actors are fine-grained, allowing data to be manipulated within the flow, instead of requiring new actors.

Tokens. Data are passed as *tokens* wrapping a single Java object (e.g. a string or an entire dataset), as well as provenance information: a trace of actors affecting the data. Tokens can assume any level of granularity: actors can receive a single token (e.g., a dataset) and emit many (e.g., data points), or vice versa, buffer tokens and emit an array (e.g., the `SequenceToArray` actor). Actors with several outputs can attach a key to each token, creating key-value pairs. Such pairs can be combined in a *container*, and actors can extract tokens by key. For instance, `MakePlotContainer` attaches 'X' and 'Y' keys so that data can be plotted.

Control actors can branch or merge sub-flows and define how data is passed between their sub-actors. `Sequence` (Sq) executes its sub-actors in sequence, passing tokens from one to the next. `Branch` (-⊏) forwards each received token to all underlying actors and executes them in parallel. `Tee` (T) splits off each input token, feeds it to its sub-flow and waits until it finishes before passing the token on. `Trigger` simply starts its sub-flow upon receiving a token (without feeding the token to its sub-flow). `Injector` passes each received token on, but also injects a new token. Some actors are conditioned on the value of the received token: `ConditionalTee` runs its sub-flow only if a stated condition holds, `If-Then-Else` runs one of two sub-workflows depending on a test, and `WhileLoop` loops over its sub-flow as long as its condition holds.

N-to-M semantics. While a tree representation cannot represent N-to-M relationships, ADAMS solves this shortcoming through *variables, key-value pairs, and global actors*. String tokens can be assigned to a variable using a `SetVariable` actor (e.g., `@{sensor}` in Fig. 1), and used as an actor parameter or reintroduced elsewhere as a token by the `Variable` actor[2]. Similarly, any object token can be stored as a key-value pair by the `SetStorageValue` actor and reintroduced by the `StorageValue` actor. Finally, actors and their sub-flows can be made global: for instance, all tokens sent to a `GlobalSink` actor are passed to the referenced global actor, as shown in Fig. 1.

Interactivity. Actors can interact with the user when needed through dialog. For instance, they can ask the user to locate an undefined file, or display a number of results and allow the user to make a subselection before proceeding.

3 Plug-In Architecture

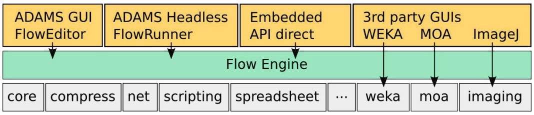


Fig. 2. ADAMS architecture

ADAMS contains an extensive library of actors enabling the inclusion of techniques from many existing libraries in a modular framework (see Fig. 2). This includes actors for machine learning techniques, importing and exporting spreadsheets, generating graphics and PDF files and sending email. A concise overview of currently supported libraries is shown in Table 1. In addition, ADAMS has a plug-in architecture to easily add new actors. A new actor can be written as a single Java class implementing a simple API. When this file is dropped into a specific folder (icon optional), ADAMS will find it and show the actor in the workflow interface. Using one of the scripting languages, actors can be developed on-the-fly without compilation.

4 Applications

ADAMS is being used in two practical applications involving large, complex workflows. First, Gas Chromatography Mass Spectrometry (GC-MS) is a technique used to detect concentrations of compounds of interest, but the raw, high-dimensional data produced is generally not amenable to processing with machine

[2] The scope of variables can be limited to one specific sub-flow by a `LocalScope` actor.

Table 1. Overview of currently supported tools, available through actors

Task	Support for
Machine learning	WEKA, MOA, parameter optimization, experiment generation
Data Streams	MOA, Twitter
Spreadsheets	MS Excel, ODF, CSV
Graphics	BMP, JPG, PNG, TIF, PDF
Imaging	ImageJ, JAI, ImageMagick, Gnuplot
Scripting	Groovy, Jython
Other	HTTP, FTP, SFTP, SSH, Email, tar/zip/bzip2/gzip

learning systems. Using ADAMS, effective data flows were designed to entirely automate this process [6]. Second, in the InfraWatch project [7], a heterogeneous sensor network of over 150 sensors is monitoring the dynamic behavior and structural health of a highway bridge. The token-based design of ADAMS proved ideal for online processing of sensor data, and its quick workflow prototyping facilitates experimentation with novel time series analysis techniques.

5 Conclusions

Most scientific workflow engines use a canvas on which operators are manually arranged and connected. While this is certainly very intuitive and appealing for many end users, it is not ideal for handling very large, complex workflows. ADAMS is a rapid prototyping workflow engine designed for researchers and practitioners dealing with large workflows. It offers a wide range of operators, a plug-in architecture to include new ones on-the-fly, and a very compact workflow representation in which operators are auto-arranged, appreciatively speeding up workflow design and maintenance. Many examples and documentation can be found on ADAMS' website: `https://adams.cms.waikato.ac.nz/`

Acknowledgements. ADAMS was developed with funding from the New Zealand Ministry of Science and Innovation (MSI), contract UOWX0707. We also thank BiG Grid, the Dutch e-Science Grid and the Netherlands Organisation for Scientific Research, NWO.

References

1. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 18, 1039–1065 (2006)
2. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2006)
3. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Data Analysis, Machine Learning and Applications, pp. 319–326 (2008)

4. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25, 528–540 (2009)
5. Bowers, S.: Scientific Workflow, Provenance, and Data Modeling Challenges and Approaches. Data Semantics 1, 19–30 (2012)
6. Holmes, G., Fletcher, D., Reutemann, P.: Predicting Polycyclic Aromatic Hydrocarbon Concentrations in Soil and Water Samples. In: Proceedings of the International Congress on Environmental Modelling and Software, IEMSS (2010)
7. Knobbe, A., Blockeel, H., Koopman, A., Calders, T., Obladen, B., Bosma, C., Galenkamp, H., Koenders, E., Kok, J.: InfraWatch: Data Management of Large Systems for Monitoring Infrastructural Performance. In: IDA Proceedings (2010)

TopicExplorer: Exploring Document Collections with Topic Models

Alexander Hinneburg, Rico Preiss, and René Schröder

Informatik, Martin-Luther-University Halle-Wittenberg, 06099 Halle, Germany
hinneburg@informatik.uni-halle.de,
{rico.preiss,rene.schroeder}@student.uni-halle.de

Abstract. The demo presents a prototype – called TopicExplorer– that combines topic modeling, key word search and visualization techniques to explore a large collection of Wikipedia documents. Topics derived by Latent Dirichlet Allocation are presented by top words. In addition, topics are accompanied by image thumbnails extracted from related Wikipedia documents to aid sense making of derived topics during browsing. Topics are shown in a linear order such that similar topics are close. Topics are mapped to color using that order. The auto-completion of search terms suggests words together with their color coded topics, which allows to explore the relation between search terms and topics. Retrieved documents are shown with color coded topics as well. Relevant documents and topics found during browsing can be put onto a shortlist. The tool can recommend further documents with respect to the average topic mixture of the shortlist.

Keywords: topic model, document browser.
URL: http://topicexplorer.informatik.uni-halle.de (Firefox 13 or later with JavaScript)

1 Introduction

The exploration of large unstructured text collections pose a difficult problem for humans in general. Search engines offer fast access to documents via keyword search, which, however, requires to know what you are searching. Therefore, search engines are not the perfect tool to explore the unknown in document collections.

Topic models may assist such an exploration by offering ordered words lists that are often recognized as general semantic topics present in the document collection. The inference algorithm estimates a set of probability distributions (topics) over the vocabulary of unique words.

A major problem is to assess the interpretability of found topics [1]. In order to develop such methods, researchers need to get an intuition about the results of topic modeling. This can be done by tracing back topics as well as document specific topic mixtures to the original documents to check how inferred topics and possible interpretations of them do appear there. Because of the sizes of

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 838–841, 2012.

a typical topic model with hundreds of topics and a document collection with thousands of documents, this is impossible to do by hand.

We set up a novel topic browser – called TopicExplorer– that helps to inspect topics and facilitates to develop and check interpretations of the found topic distributions. For these purposes, TopicExplorer combines topic modeling, key word search and visualization to give the user an intuition about the strengths and drawbacks of the computed topics.

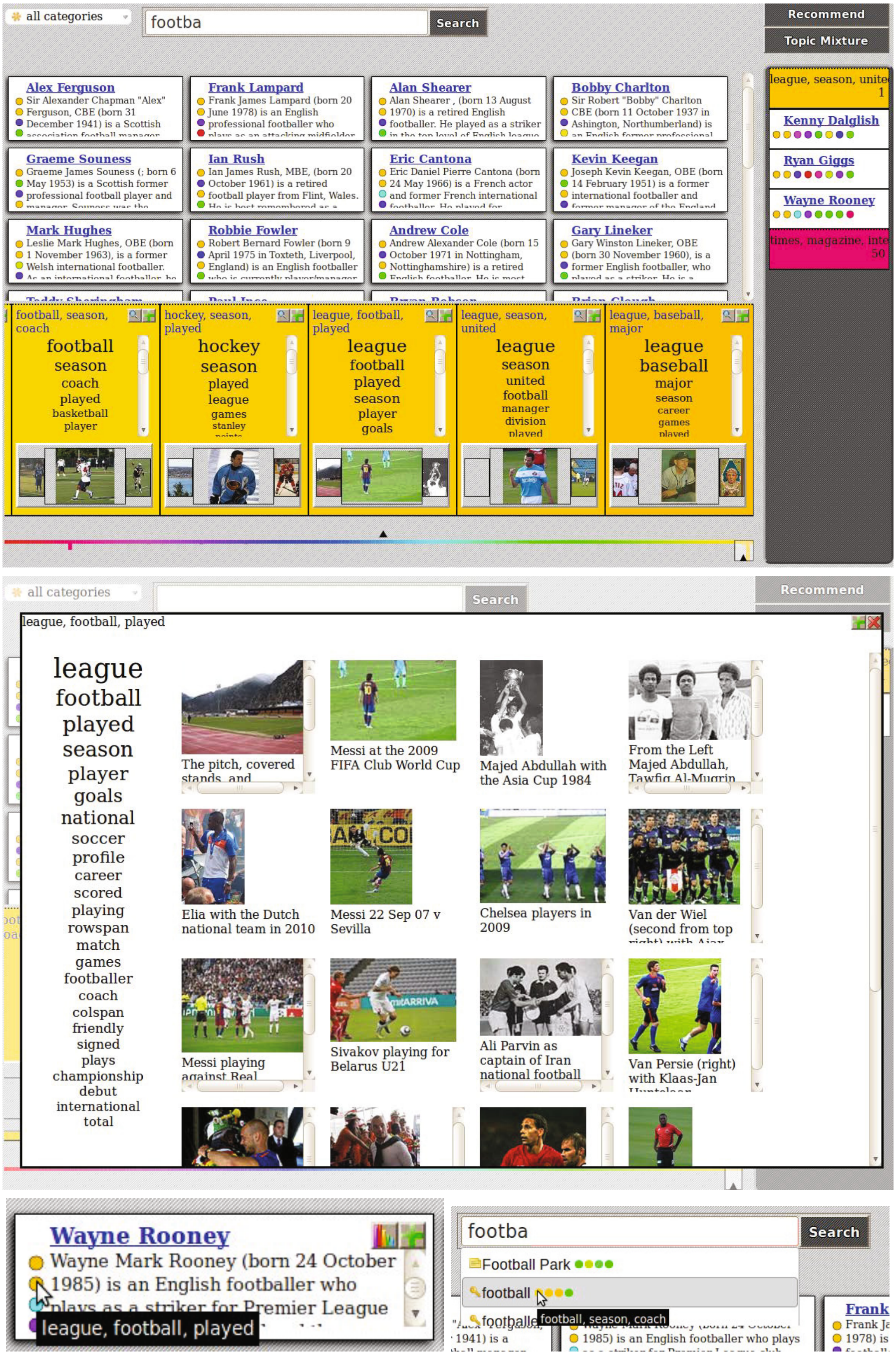
2 Demonstration of TopicExplorer

The underlying data for our demonstration is the subset of the English Wikipedia that contains all pages in the general categories person or place. Without page forwards, the document collection contains 716,874 documents in total. We build an LDA topic model [2] using MALLET [3] having 200 topics.

TopicExplorer is a web-based application with a MySQL back-end that stores data of the document collection as well as the topic model. The GUI (see Figure 1 top) shows all topics as a horizontal row of colored boxes at the bottom. Each box shows the list of most important words of a topic that are ordered by decreasing probability of a word given a topic $p(w|z)$. The font size is also proportional to this probability and indicates how fast the probabilities are decreasing. In addition to the words, each topic in accompanied by a list of associated images that can be scrolled by clicking on the smaller left or right thumbnails. Images are associated to a topic z if many words in the neighborhood of the image link in the wiki-text are probabilistically assigned to z. Top words as well as images of a topic are shown as full screen overlay when the lens icon of topic is clicked (see Figure 1 middle). Viewing images and words together helps to find an interpretation of the topic.

The topics are ordered by similarity into a linear list that can be horizontally scrolled using the slider (gray box) below the color bar at the very bottom of the GUI. The ordering is derived using Wards hierarchical clustering as implemented in R that takes cosine similarities the between topic distributions as input and outputs a layout of the dendrogram. That layout implies an ordering of the leafs, which are the topics in this case. The ordering is quite helpful to organize the large topic set, e.g. the screen-shot (Figure 1 top) shows different team sports (football, hockey, soccer and baseball) close together. The linear order of topics is used to map topics to colors (the rainbow color-map in this demo). Topic color is used to indicate the most important topics of a document in the respective snippets. By hovering with the mouse over such a colored circle shows a tool tip with the three most important words, e.g. Figure 1 bottom left. Clicking a circle takes the user directly to the topic. A similar visualization is used to indicate the most important topics of a word in the auto-completion when typing a search keyword (see Figure 1 bottom right). There, topics are ordered by decreasing $p(z|w)$. The visualization in the autocompletion helps to identify ambiguous words, like football.

Topics as well as documents can be added to a shortlist at the right of the screen (Figure 1 top). Using the Recommend-Button, the average topic mixture



Alex Ferguson
Sir Alexander Chapman "Alex" Ferguson, CBE (born 31 December 1941) is a Scottish

Frank Lampard
Frank James Lampard (born 20 June 1978) is an English professional footballer who

Alan Shearer
Alan Shearer , (born 13 August 1970) is a retired English footballer. He played as a striker

Bobby Charlton
Sir Robert "Bobby" Charlton CBE (born 11 October 1937 in Ashington, Northumberland) is

Graeme Souness
Graeme James Souness (; born 6 May 1953) is a Scottish former professional football player and

Ian Rush
Ian James Rush, MBE, (born 20 October 1961) is a retired football player from Flint, Wales.

Eric Cantona
Eric Daniel Pierre Cantona (born 24 May 1966) is a French actor and former French international

Kevin Keegan
Joseph Kevin Keegan, OBE (born 14 February 1951) is a former international footballer and

Mark Hughes
Leslie Mark Hughes, OBE (born 1 November 1963), is a former Welsh international footballer.

Robbie Fowler
Robert Bernard Fowler (born 9 April 1975 in Toxteth, Liverpool, England) is an English footballer

Andrew Cole
Andrew Alexander Cole (born 15 October 1971 in Nottingham, Nottinghamshire) is a retired

Gary Lineker
Gary Winston Lineker, OBE (born 30 November 1960), is a former English footballer, who

The pitch, covered stands, and

Messi at the 2009 FIFA Club World Cup

Majed Abdullah with the Asia Cup 1984

From the Left Majed Abdullah, Tawfiq Al-Muqrin

Elia with the Dutch national team in 2010

Messi 22 Sep 07 v Sevilla

Chelsea players in 2009

Van der Wiel (second from top right) with Ajax

Messi playing against Real

Sivakov playing for Belarus U21

Ali Parvin as captain of Iran national football

Van Persie (right) with Klaas-Jan Huntelaar

Wayne Rooney
Wayne Mark Rooney (born 24 October 1985) is an English footballer who plays as a striker for Premier League

1985) is an English footballer who plays

Frank Ja
1978) is

Fig. 1. TopicExplorer

distribution of topics and documents in the shortlist is used as a cosine similarity search query to find documents with similar topic mixture distribution. We use LSH [4] to make this search of the whole document collection fast enough for an interactive application. This similarity search helps to trace the topics back to the original documents that define the topic. Keyword search completes the GUI, which helps the user to focus the evaluation of the topic model onto a specific area of interest.

3 Related Work

Related software projects include the Topic Model Visualization Engine by Chaney and Blei (`http://code.google.com/p/tmve/`) that transforms a topic model into a precomputed set of web pages. It shows important words for each topic as well as related documents and related topics. The Topical Guide (`https://facwiki.cs.byu.edu/nlp/index.php/Topical_Guide`) shows similar informations for each topic. In addition, it allows to filter by topics, different metrics, words and documents and can produce parallel coordinates plots that relate topics with other meta-data. The closest match to our TopicExplorer is the topic-based search interface SearchInaBox to Wikipedia by Buntine (`http://wikipedia.hiit.fi/H100topiclist.html`). It allows to search documents by keywords and filters optionally afterwards by topical text. The result list shows the important topics of the documents. Additionally to the search engine functionality, it offers a topic browser that shows important words and documents for each topic.

Our TopicExplorer is different from all these tools by showing images related to topics to aid intuitive interpretation of topics. Furthermore, it is the only one that allows similarity search of the document collection by arbitrary topic mixture distributions.

Acknowledgements. We thank R. Fildebrandt, T. Gottlieb, T. Nguyen, M. Pfuhl, M. Kunze and C. Wangemann for their contributions to the code base of TopicExplorer as well as A. Both of Unister GmbH for helpful discussions.

References

1. Blei, D.M.: Probabilistic topic models. Commun. ACM 55(4), 77–84 (2012)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022 (2003)
3. McCallum, A.K.: Mallet: A machine learning for language toolkit (2002), `http://mallet.cs.umass.edu`
4. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Proc. of STOC 2002, pp. 380–388. ACM (2002)

VIKAMINE – Open-Source Subgroup Discovery, Pattern Mining, and Analytics

Martin Atzmueller[1] and Florian Lemmerich[2]

[1] University of Kassel,
Knowledge and Data Engineering Group
Wilhelmshöher Allee 73, 34121 Kassel, Germany
[2] University of Wuerzburg,
Artificial Intelligence and Applied Informatics Group,
Am Hubland, 97074 Wuerzburg, Germany
atzmueller@cs.uni-kassel.de,
lemmerich@informatik.uni-wuerzburg.de

Abstract. This paper presents an overview on the VIKAMINE[1] system for subgroup discovery, pattern mining and analytics. As of VIKAMINE version 2, it is implemented as rich-client platform (RCP) application, based on the Eclipse[2] framework. This provides for a highly-configurable environment, and allows modular extensions using plugins. We present the system, briefly discuss exemplary plugins, and provide a sketch of successful applications.

Keywords: Pattern Mining, Subgroup Discovery, Analytics, Open-Source.

1 VIKAMINE

Subgroup discovery and pattern mining are important descriptive data mining tasks. They can be applied, for example, in order to obtain an overview on the relations in the data, for automatic hypotheses generation, and for a number of knowledge discovery applications. We present the VIKAMINE system for such applications.

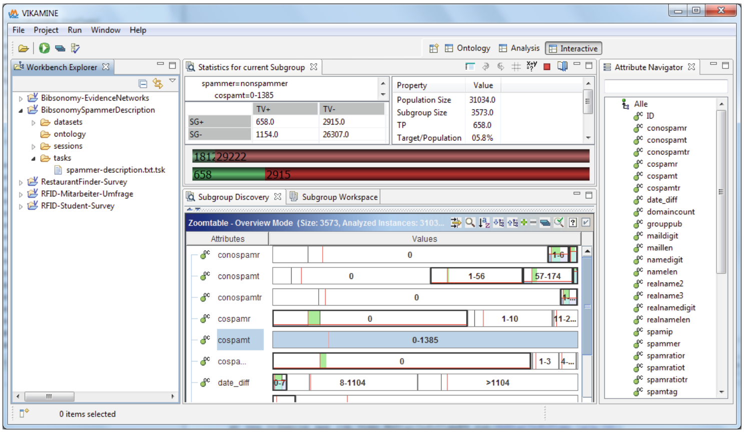
VIKAMINE is targeted at a broad range of users, from industrial practitioners to ML/KDD researchers, students, and users interested in knowledge discovery and data analysis in general. It features a variety of state-of-the-art automatic algorithms, visualizations, broad extensibility, and rich customization capabilities enabled by the Eclipse RCP environment. In contrast to general purpose data mining systems, it is specialized for the task of subgroup discovery and pattern mining. It focuses on visual, interactive and knowledge-intensive methods and aims to integrate a distinctive set of features with an easy-to-use interface:

- **State-of-the-Art Algorithms:** VIKAMINE comes with a variety of established and state-of-the-art algorithms for automatic subgroup discovery, e.g., Beam-Search [7], BSD [9], and SD-Map* [2]. A wide variety of popular interestingness measures can be used for binary, nominal, and numeric target concepts.

[1] http://www.vikamine.org
[2] http://www.eclipse.org

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 842–845, 2012.

- **Visualizations:** For successful interactive mining, specialized visualizations are essential to achieve a quick and intuitive understanding of the data and mined patterns [6]. Visualizations implemented in VIKAMINE include, for example, the *zoomtable* [5] for visual and semi-automatic subgroup discovery shown in Figure 1, pattern specialization graphs, or visualizations of patterns in the ROC-space.



spammer=nonspammer
cospamt=0-1385

	TV+	TV-
SG+	658.0	2915.0
SG-	1154.0	26307.0

Property	Value
Population Size	31034.0
Subgroup Size	3573.0
TP	658.0
Target/Population	05.8%

Zoomtable - Overview Mode (Size: 3573, Analyzed Instances: 3103...

Attributes	Values
conospamr	0 | 1-6
conospamt	0 | 1-56 | 57-174
conospamtr	0 | 1-...
cospamr	0 | 1-10 | 11-2...
cospamt	0-1385
cospa...	0 | 1-3 | 4-...
date_diff	0-7 | 8-1104 | >1104

Fig. 1. Screenshot of the VIKAMINE workbench: Projects (left), the zoomtable (middle, bottom), pattern statistics (middle, top) and the attribute view (right)

- **Prior Knowledge:** VIKAMINE supports various methods to integrate prior knowledge into the mining process, e.g., considering expected/known dependencies, causal analysis, and pattern filtering options. Background knowledge can be acquired using form-based approaches or text documents.
- **Extensibility:** Using the Rich Client Platform of Eclipse, VIKAMINE can easily be extended by specialized plug-ins for the target application area. Customized extension points allow, for example, for a quick integration of new interestingness measures, search algorithms, visualizations, types of background knowledge and domain specific views on the data. Specialized plugins using such extension points also allow for the integration of other data mining and statistic libraries, e.g., for a connection to the statistic environment R[3].
- **Modularity:** VIKAMINE utilizes a strict separation between kernel components, i.e., data representations and algorithms, and the graphical interface components. Thus, the algorithmic core functionalities of VIKAMINE can be easily integrated in other systems and applications, e.g., for the integration in production environments, or for evaluations of algorithms by researchers.
- **Organization:** Automatic discovery tasks can declaratively be stored in XML. By utilizing Eclipse workspace and project concepts, VIKAMINE supports the user in keeping track of all the data, performed tasks and results of a data mining project.

[3] http://www.r-project.org/

2 Selected Plugins

- **VIKAMINE R-Plugin**: In order to integrate external data mining and analysis methods, the VIKAMINE R-Plugin features the ability to connect to the R[4] environment for statistical computing. Using the plugin, for example, external methods and visualizations can easily be integrated using R scripts.
- **VIKAMINE Geo-Plugin**: For mining spatial data, the VIKAMINE Geo-Plugin provides for specialized mining and visualization options taking geo-locations into account. For example, patterns characterizing specific locations can be mined, including tagging data for the description [8]. The plugin provides suitable visualization options for further inspection, browsing, and refinement.
- **VIKAMINE Community Mining Plugin**: Descriptive community mining solves one of the major problems of standard approaches, i.e., that the discovered communities have no inherent *meaning*. The community mining plugin implements such an approach; using a graph structure and descriptive information, e.g., friendship networks and tags applied by the users [4], descriptive patterns with high commnity qualities according to standard measures are obtained.

3 Exemplary Applications

- **Medical Knowledge Discovery and Quality Control**: VIKAMINE has been applied for large-scale knowledge discovery and quality control in the clinical application SONOCONSULT , cf., [10]. According to the physicians, subgroup discovery and analysis is quite suitable for examining common medical questions, e.g. whether a certain pathological state is significantly more frequent if combinations of other pathological states exist or if there are diagnoses, which one physician documents significantly more or less frequently than the average. Furthermore, VIKAMINE also provides an intuitive interface for providing an overview on the data, in addition to the knowledge discovery and quality control functions.
- **Industrial Fault Analysis**: Another application concerned large-scale technical fault analysis. The task required the identification of subgroups (as combination of certain factors) that cause a significant increase/decrease in the fault/repair rates of certain products. In the application, one important goal was the identification of subgroups (as combination of certain factors) that cause a significant increase/ in certain parameters. This concerns, for example, the number of service requests for a certain technical component, the fault/repair rate of a certain manufactured product, or the number of calls of customers to service support. Such applications often require the utilization of continuous parameters. Then, the target concepts can often not be analyzed sufficiently using the standard discretization techniques, since the discretization of the variables causes a loss of information. In this context, VIKAMINE provides state-of-the-art algorithmic implementations, cf. [2], for supporting the knowledge discovery and analysis, and enables an effective involvement of the domain experts.

[4] http://www.r-project.org

- **Mining Social Media**: In addition to the approaches sketched above VIKAMINE features a number of successful applications in the social media domain, see [8,4]. It was applied, for example, for obtaining descriptive profiles of spammers, i.e., for their characterization [3]. The mined patterns capturing certain spammer subgroups provide explanations and justifications for marking or resolving spammer candidates in a social bookmarking systems. In such contexts, it is also useful to identify high-quality tags, i.e., tags with a certain maturity, cf. [1]. VIKAMINE was applied for obtaining maturity profiles of tags based on graph centrality features on the tag—tag cooccurrance graph. Then, the obtained information can be utilized for tag recommendations, faceted browsing, or for improving search.

4 Conclusions

In this paper, we presented an overview on VIKAMINE, focusing on efficient and effective pattern mining and subgroup discovery. As of version 2, VIKAMINE is implemented as an Eclipse-based rich-client platform (RCP) application. This provides for an integrated system that is highly modular and broadly extensible using plugins. VIKAMINE can be freely downloaded from `http://www.vikamine.org` under an *LGPL* open-source license.

References

1. Atzmueller, M., Benz, D., Hotho, A., Stumme, G.: Towards Mining Semantic Maturity in Social Bookmarking Systems. In: Proc. Workshop Social Data on the Web, International Semantic Web Conference (2011)
2. Atzmueller, M., Lemmerich, F.: Fast Subgroup Discovery for Continuous Target Concepts. In: Proc. 18th Intl. Symp. Method. Intelligent Systems. Springer (2009)
3. Atzmueller, M., Lemmerich, F., Krause, B., Hotho, A.: Who are the Spammers? Understandable Local Patterns for Concept Description. In: Proc. 7th Conference on Computer Methods and Systems (2009)
4. Atzmueller, M., Mitzlaff, F.: Efficient descriptive community mining. In: Proc. 24th Intl. FLAIRS Conference. AAAI Press (2011)
5. Atzmueller, M., Puppe, F.: Semi-Automatic Visual Subgroup Mining using VIKAMINE. Journal of Universal Computer Science, Visual Data Mining 11(11), 1752–1765 (2005)
6. Gamberger, D., Lavrac, N., Wettschereck, D.: Subgroup Visualization: A Method and Application in Population Screening. In: Proc. 7th Intl. Workshop on Intelligent Data Analysis in Medicine and Pharmacology, IDAMAP 2002 (2002)
7. Lavrac, N., Kavsek, B., Flach, P., Todorovski, L.: Subgroup Discovery with CN2-SD. Journal of Machine Learning Research 5, 153–188 (2004)
8. Lemmerich, F., Atzmueller, M.: Modeling Location-based Profiles of Social Image Media using Explorative Pattern Mining. In: Proc. IEEE SocialCom 2011, Workshop on Modeling Social Media (MSM 2011). IEEE Computer Society, Boston (2011)
9. Lemmerich, F., Rohlfs, M., Atzmueller, M.: Fast Discovery of Relevant Subgroup Patterns. In: Proc. 21st Intl. FLAIRS Conference, pp. 428–433. AAAI Press (2010)
10. Puppe, F., Atzmueller, M., Buscher, G., Huettig, M., Lührs, H., Buscher, H.-P.: Application and Evaluation of a Medical Knowledge-System in Sonography (SonoConsult). In: Proc. 18th European Conference on Artificial Intelligence, pp. 683–687 (2008)

Learning Submodular Functions[*]

Maria-Florina Balcan[1] and Nicholas J.A. Harvey[2]

[1] Georgia Institute of Technology, School of Computer Science
[2] University of British Columbia

Abstract. Submodular functions are discrete functions that model laws of diminishing returns and enjoy numerous algorithmic applications that have been used in many areas, including combinatorial optimization, machine learning, and economics. In this work we use a learning theoretic angle for studying submodular functions. We provide algorithms for learning submodular functions, as well as lower bounds on their learnability. In doing so, we uncover several novel structural results revealing both extremal properties as well as regularities of submodular functions, of interest to many areas.

Submodular functions are a discrete analog of convex functions that enjoy numerous applications and have structural properties that can be exploited algorithmically. They arise naturally in the study of graphs, matroids, covering problems, facility location problems, etc., and they have been extensively studied in operations research and combinatorial optimization for many years [8]. More recently submodular functions have become key concepts both in the machine learning and algorithmic game theory communities. For example, submodular functions have been used to model bidders' valuation functions in combinatorial auctions [12,6,3,14], and for solving feature selection problems in graphical models [11] or for solving various clustering problems [13]. In fact, submodularity has been the topic of several tutorials and workshops at recent major conferences in machine learning [1,9,10,2].

Despite the increased interest on submodularity in machine learning, little is known about the topic from a learning theory perspective. In this work, we provide a statistical and computational theory of learning submodular functions in a distributional learning setting.

Our study has multiple motivations. From a foundational perspective, submodular functions are a powerful, broad class of important functions, so studying their learnability allows us to understand their structure in a new way. To draw a parallel to the Boolean-valued case, a class of comparable breadth would be the class of monotone Boolean functions; the learnability of such functions has been intensively studied [4,5]. From an applications perspective, algorithms for learning submodular functions may be useful in some of the applications where these functions arise. For example, in the context of algorithmic game theory

[*] This note summarizes several results in the paper "Learning Submodular Functions", by Maria Florina Balcan and Nicholas Harvey, which appeared The 43rd ACM Symposium on Theory of Computing (STOC) 2011.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 846–849, 2012.

and economics, an auctioneer may use such an algorithm to sketch the players' valuation functions before designing the auction, companies might want to learn the valuation functions of their customers to in order to predict demand, etc. More broadly, the problem of learning submodular functions is natural for a wide range of settings where one would like to predict the value of some function over objects described by features, where the features have positive but decreasing marginal impact on the function's value. Examples include predicting the rate of growth of jobs in cities as a function of various amenities or enticements that the city offers, predicting the sales price of a house as a function of features (such as an updated kitchen, hardwood floors, extra bedrooms, etc.) that it might have, and predicting the demand for a new laptop as a function of various add-ons which might be included. In all of these settings (and many others) it is natural to assume diminishing returns, making them well-suited to a formulation as a problem of learning a submodular function.

To study the learnability of submodular functions, we introduce a learning model for approximate distributional learning, which can be described as follows. There is an underlying, fixed but unknown distribution over the subsets of the ground set and a fixed but unknown submodular target function f^*, and the goal is to algorithmically provide a good approximation of the target function with respect to the underlying distribution, in polynomial time, based on a polynomial number of samples from the underlying distribution. Formally, the goal is to output a hypothesis function f that, with probability $1 - \delta$ over the choice of examples, is a good approximation of the target f^* on most of the points coming from D. Here "most" means a $1-\epsilon$ fraction and "good approximation" means that $f(S) \leq f^*(S) \leq \alpha \cdot f(S)$ for some approximation factor α. Our results on learning submodular functions are presented in this new model, which we call the *PMAC model*; this abbreviation stands for "Probably Mostly Approximately Correct". Note that this learning model differs from the usual PAC-learning model. In our model, one must *approximate* the value of a function on a set of large measure, with high confidence. In contrast, the traditional PAC-learning model usually studies learnability of much simpler classes of Boolean functions. There, one must compute the value *exactly* on a set of large measure, with high confidence.

We prove nearly matching $\alpha = O(n^{1/2})$ upper and $\alpha = \tilde{\Omega}(n^{1/3})$ lower bounds on the approximation factor achievable when the algorithm receives only $poly(n, 1/\epsilon, 1/\delta)$ examples from an arbitrary (fixed but unknown distribution). We additionally provide a better constant approximation factor learning algorithm for the case where the underlying distribution is a product distribution, which is based on a new result showing a strong concentration of submodular functions.

We start by showing that it is possible to PMAC-learn the general class of non-negative, monotone submodular functions with an approximation factor of $\sqrt{n+1}$. To prove this we use a structural result in [7] which shows that any monotone, non-negative submodular function can be approximated within a factor of $\sqrt{n+1}$ on every point by the square root of an additive function. Using this result, we show how to convert the problem of learning a submodular function in the

PMAC model to the problem of learning a linear separator in R^{n+1} in the usual PAC model. We remark that an improved structural result for any subclass of submodular functions immediately implies an improved analysis of our algorithm for that subclass.

We introduce a new family of matroids to show a comparable lower bound: any algorithm that uses a polynomial number of examples cannot PMAC-learn the class of submodular functions with an approximation factor $o(n^{1/3}/\log n)$. In fact, we show that even *weak* PMAC-learning is not possible — any algorithm can do only negligibly better than random guessing for this class of functions. Moreover, this lower bound holds even if the algorithm is told the underlying distribution and it is given the ability to query the function on inputs of its choice and even if the queries are adaptive. In other words this lower bound holds even in the PMAC model augmented with value queries.

This lower bound holds even for matroid rank functions, but it uses a distribution on inputs which is a non-product distribution. It turns out that the use of such a distribution is necessary: using Talagrand's inequality, we prove that a constant approximation factor can be achieved for matroid rank functions under product distributions.

To prove the lower bound, we consider the following technical problem. We would like find an injective map $\rho : \{0,1\}^d \to \{0,1\}^n$ and real numbers $\alpha \ll \beta$ such that every Boolean function f on $\{0,1\}^d$ can be mapped to a non-negative, monotone, submodular function $\tilde{f}$ on $\{0,1\}^n$ satisfying $f(x) = 0 \Rightarrow \tilde{f}(\rho(x)) \leq \alpha$ and $f(x) = 1 \Rightarrow \tilde{f}(\rho(x)) \geq \beta$. This implies a lower bound on learning submodular functions with approximation factor $\frac{\beta}{\alpha}$ when $d = \omega(\log n)$. A trivial construction is obtained using partition matroids, with $\alpha = 0$, $\beta \leq \frac{n}{2}$ and $d \leq \log(\lfloor n/\beta \rfloor)$; here d is too small to be of interest. Another easy construction is obtained using paving matroids, with $\alpha = \frac{n}{2}$, $\beta = \frac{n}{2} + 1$, and any $d = n - \Omega(\log^4(n))$; here d is large, but there is only a small additive gap between α and β. Our new family of matroids is a common generalization of partition and paving matroids. We use them to obtain a construction with $\alpha = 16d$, $\beta = n^{1/3}$ and any $d = o(n^{1/3})$; this gives a large multiplicative gap between α and β.

Our work has several interesting by-products. One is the PMAC-learning model, which studies both the probability mass of points on which the hypothesis does well and the multiplicative approximation achieved on those points. Another by-product of our work is our new family of matroids which reveals interesting extremal properties of submodular functions. Roughly speaking, we show that a small Boolean cube can be embedded into a large Boolean cube so that any $\{0,1\}$-valued function on the small cube maps to a function that is submodular on the large cube but is now $\{\alpha, \beta\}$-valued with $\alpha \ll \beta$ (on the points to which the small cube was embedded).

References

1. NIPS workshop on discrete optimization in machine learning: Submodularity, sparsity & polyhedra, DISCML (2009), http://www.discml.cc/
2. NIPS workshop on discrete optimization in machine learning (discml): Uncertainty, generalization and feedback (2011), http://las.ethz.ch/discml/

3. Balcan, M.F., Blum, A., Mansour, Y.: Item pricing for revenue maxmimization. In: ACM Conference on Electronic Commerce (2009)
4. Blum, A., Burch, C., Langford, J.: On learning monotone boolean functions. In: FOCS (1998)
5. Dachman-Soled, D., Lee, H.K., Malkin, T., Servedio, R.A., Wan, A., Wee, H.M.: Optimal Cryptographic Hardness of Learning Monotone Functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 36–47. Springer, Heidelberg (2008)
6. Dobzinski, S., Nisan, N., Schapira, M.: Truthful Randomized Mechanisms for Combinatorial Auctions. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, pp. 644–652 (2006)
7. Goemans, M., Harvey, N., Iwata, S., Mirrokni, V.: Approximating submodular functions everywhere. In: ACM-SIAM Symposium on Discrete Algorithms (2009)
8. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer (1993)
9. Krause, A., Guestrin, C.: Beyond convexity: Submodularity in machine learning (2008), http://www.select.cs.cmu.edu/tutorials/icml08submodularity.html
10. Krause, A., Guestrin, C.: Intelligent information gathering and submodular function optimization (2009), http://submodularity.org/ijcai09/index.html
11. Krause, A., Guestrin, C.: Near-optimal nonmyopic value of information in graphical models. In: UAI (2005)
12. Lehmann, B., Lehmann, D.J., Nisan, N.: Combinatorial auctions with decreasing marginal utilities. Games and Economic Behavior 55, 270–296 (2006)
13. Narasimhan, M., Bilmes, J.: Local search for balanced submodular clusterings. In: Twentieth International Joint Conference on Artificial Intelligence (2007)
14. Vondrák, J.: Optimal approximation for the submodular welfare problem in the value oracle model. In: STOC (2008)

Matrix Factorization as Search[⋆]

Kristian Kersting[1,2], Christian Bauckhage[1],
Christian Thurau[3], and Mirwaes Wahabzada[1]

[1] Fraunhofer IAIS, Schloss Birlinghoven, Sankt Augustin, Germany
[2] Institute of Geodesy and Geoinformation, University of Bonn, Germany
[3] Game Analytics Aps., Copenhagen, Denmark

Abstract. Simplex Volume Maximization (SiVM) exploits distance geometry for efficiently factorizing gigantic matrices. It was proven successful in game, social media, and plant mining. Here, we review the distance geometry approach and argue that it generally suggests to factorize gigantic matrices using search-based instead of optimization techniques.

1 Interpretable Matrix Factorization

Many modern data sets are available in form of a real-valued $m \times n$ matrix $\mathbf{V}$ of rank $r \leq \min(m, n)$. The columns $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of such a data matrix encode information about n objects each of which is characterized by m features. Typical examples of objects include text documents, digital images, genomes, stocks, or social groups. Examples of corresponding features are measurements such as term frequency counts, intensity gradient magnitudes, or incidence relations among the nodes of a graph. In most modern settings, the dimensions of the data matrix are large so that it is useful to determine a compressed representation that may be easier to analyze and interpret in light of domain-specific knowledge. Formally, compressing a data matrix $\mathbf{V} \in \mathbb{R}^{m \times n}$ can be cast as a *matrix factorization* (MF) task. The idea is to determine factor matrices $\mathbf{W} \in \mathbb{R}^{m \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times n}$ whose product is a low-rank approximation of $\mathbf{V}$. Formally, this amounts to a minimization problem $\min_{\mathbf{W},\,\mathbf{H}} \left\|\mathbf{V} - \mathbf{WH}\right\|^2$ where $\|\cdot\|$ denotes a suitable matrix norm, and one typically assumes $k \ll r$.

A common way of obtaining a low-rank approximation stems from truncating the singular value decomposition (SVD) where $\mathbf{V} = \mathbf{WSU}^T = \mathbf{WH}$. The SVD is popular for it can be solved analytically and has significant statistical properties. The column vectors $\mathbf{w}_i$ of $\mathbf{W}$ are orthogonal basis vectors that coincide with the directions of largest variance in the data. Although there are many successful applications of the SVD, for instance in information retrieval, it has been criticized because the $\mathbf{w}_i$ may lack interpretability with respect to the field from which the data are drawn [6]. For example, the $\mathbf{w}_i$ may point in the direction of negative orthants even though the data itself is strictly non-negative. Nevertheless, data analysts are often tempted to reify, i.e., to assign a "physical"

[⋆] The authors would like to thank the anonymous reviewers for their comments. This work was partly supported by the Fraunhofer ATTRACT fellowship STREAM.

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 850–853, 2012.

meaning or interpretation to large singular components. In most cases, however, this is not valid. Even if reification is justified, the interpretative claim cannot arise from mathematics, but must be based on an intimate knowledge of the application domain.

The most common way of compressing a data matrix such that the resulting basis vectors are interpretable and faithful to the data at hand is to impose additional constraints on the matrices $\mathbf{W}$ and $\mathbf{H}$. An example is *non-negative* MF (NMF), which imposes the constraint that entries of $\mathbf{W}$ and $\mathbf{H}$ are non-negative. Another example of a constrained MF method is *archetypal analysis* (AA) as introduced by [3]. It considers the NMF problem where $\mathbf{W} \in \mathbb{R}^{n \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times n}$ are additionally required to be column stochastic matrices, i.e., they are to be non-negative and each of their columns is to sum to 1. AA therefore represents every column vector in $\mathbf{V}$ as a convex combination of convex combinations of a *subset of the columns* of $\mathbf{V}$. Such constrained MF problems are traditionally solved analytically since they constitute quadratic optimization problems. Although they are convex in either $\mathbf{W}$ or $\mathbf{H}$, they are however not convex in $\mathbf{WH}$ so that we suffers from many local minima. Moreover, their memory and runtime requirements scale quadratically with the number n of data and therefore cannot easily cope with modern large-scale problems. A recent attempt to circumvent these problems is the CUR decomposition [6]. It aims at minimizing $\|\mathbf{V} - \mathbf{CUR}\|^2$ where the columns of $\mathbf{C}$ are selected from the columns of $\mathbf{V}$, the rows of $\mathbf{R}$ are selected from the rows of $\mathbf{V}$, and $\mathbf{U}$ contains scaling coefficients. Similar to AA, the factorization is expressed in terms of actual data elements and hence is readily interpretable. However, in contrast to AA, the selection is not determined analytically but by means of importance sampling from the data at hand. While this reduces memory and runtime requirements, it still requires a complete view of the data. Therefore, neither of the methods discussed so far easily applies to growing dataset that nowadays become increasingly common.

2 Matrix Factorization as Search

MF by means of column subset selection allows one to cast MF as a *volume maximization problem* rather than as norm minimization [2]. It can be shown that a subset $\mathbf{W}$ of k columns of $\mathbf{V}$ yields a better factorization than any other subset of size k, if the volume of the parallelepiped spanned by the columns of $\mathbf{W}$ exceeds the volumes spanned by the other selections. Following this line, we have recently proposed a linear time approximation for maximising the volume of the simplex $\Delta\mathbf{W}$ whose vertices correspond to the selected columns [9]. Intuitively, we aim at approximating the data by means of convex combinations of selected vectors $\mathbf{W} \subset \mathbf{V}$. That is, we aim at compressing the data such that $\mathbf{v}_i \approx \sum_{j=1}^{k} \mathbf{w}_j \, h_{ji}$ where $\mathbf{h}_i \succeq \mathbf{0} \;\wedge\; \mathbf{1}^T\mathbf{h}_i = 1 \;\; \forall i$. Then, data vectors situated on the inside of the simplex $\Delta\mathbf{W}$ can be reconstructed perfectly, i.e., $\|\mathbf{v}_i - \mathbf{W}\mathbf{h}_i\|^2 = 0$. Accordingly, the larger the volume of $\Delta\mathbf{W}$, the better the corresponding low-rank approximation of the entire data set will be. Such volume maximization approaches are more efficient than methods based on minimizing a matrix norm. Whereas the latter requires computing both matrices $\mathbf{W}$ and $\mathbf{H}$ in every iteration, volume maximization methods compute the coefficient matrix $\mathbf{H}$ only after the matrix of basis vectors $\mathbf{W}$ has been determined. Moreover, whereas evaluating $\|\mathbf{V} - \mathbf{WH}\|^2$ is of complexity $O(n)$ for n data points $\mathbf{v}_i$, evaluating $\mathrm{Vol}(\mathbf{W})$ or $\mathrm{Vol}(\mathit{\Delta}\mathbf{W})$ requires $O(k^3)$ for the $k \ll n$ currently selected columns. Moreover, transferring volume maximization from parallelepipeds to simplices has the added benefit that it allows for the use of *distance geometry.* Given the lengths $d_{i,j}$ of the edges between the k vertices of a $(k-1)$-simplex $\mathit{\Delta}\mathbf{W}$, its volume $\mathrm{Vol}^k_{\mathit{\Delta}\mathbf{W}}$ can be computed based on this distance information only **(*)**: $\mathrm{Vol}^k_{\mathit{\Delta}\mathbf{W}} = \sqrt{\frac{-1^k}{2^{k-1}\big((k-1)!\big)^2}\det\big(\mathbf{A}\big)}$ where $\det\big(\mathbf{A}\big)$ is the *Cayley-Menger* determinant [1]. And, it naturally leads to search-based MF approaches.

A simple greedy best-first search algorithm for MF that immediately follows from what has been discussed so far works as follows. Given a data matrix $\mathbf{V}$, we determine an initial selection $X_2 = \{a, b\}$ where $\mathbf{v}_a$ and $\mathbf{v}_b$ are the two columns that are maximally far apart. That is, we initialize with the largest possible 1-simplex. Then, we consider every possible extension of this simplex by another vertex and apply **(*)** to compute the corresponding volume Vol'. The extended simplex that yields the largest volume is considered for further expansion. This process continues, until k columns have been selected from $\mathbf{V}$. Lower bounding **(*)** by assuming that all selected vertices are equidistant turns this greedy best-first into the linear time MF approach called Simplex Volume Maximization (SiVM) [9]. SiVM was proven to be successful for the fast and interpretable analysis of massive game and twitter data [7], of large, sparse graphs [8] as well as — when combined with statistical learning techniques — of drought stress of plants [4,5]. However, we can explore and exploit the link established between MF and search even further. For instance, a greedy stochastic hill climbing algorithm (sSiVM) starts with a random initial selection of k columns of $\mathbf{V}$ and iteratively improves on it. In each iteration, a new candidate column is chosen at random and tested against the current selection: for each of the currently selected columns, we verify if replacing it by the new candidate would increase the simplex volume according to **(*)**. The column whose replacement results in the largest gain is replaced. An apparent benefit of sSiVM is that it does not require batch processing or knowledge of the entire data matrix. It allows for timely data matrix compression even if the data arrive one at a time. Since it consumes only $O(k)$ memory, it represents a truly low-cost approach to MF.

In an ongoing project on social media usage, we are running a script that constantly downloads user annotated images from the Internet. We are thus in need of a method that allows for compressing this huge collection of data in an online fashion. sSiVM appears to provide a solution. To illustrate this, we considered a standard data matrix representing Internet images collected by [10]. This publicly available data has the images re-scaled to a resolution of 32×32 pixels in 3 color channels and also provides an abstract representation using 384-dimensional GIST feature vectors. Up to when writing the present paper, sSiVM processed a stream of about 1,600,000 images (randomly selected). This

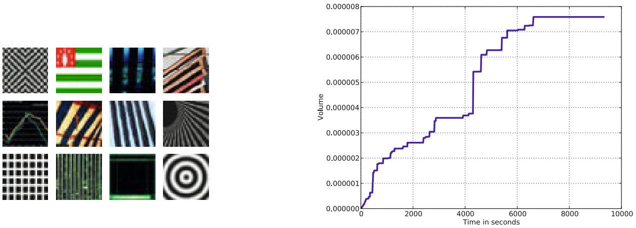


Fig. 1. (Left) Examples of 12 basis images found after 1,6 million Internet images were seen by sSiVM. (Right) Temporal evolution of the solution produced by sSiVM while computing the results shown on the left-hand side.

amounts to a matrix of 614,400,000 entries. Except for sSiVM, none of the methods discussed in this paper could reasonably handle this setting when running on a single computer. Figure 1(Left) shows a selection of 12 *basis images* obtained by sSiVM. They bear a geometric similarity to Fourier basis functions or Gabor filters. This is in fact a convincing sanity check, since GIST features are a frequency domain representation of digital images; images most similar to elementary sine or cosine functions form the extreme points in this space. Together with the measured runtime, see Fig. 1(Right), these results underline that search-based MF approaches are a viable alternative to optimization approaches.

References

1. Blumenthal, L.M.: Theory and Applications of Distance Geometry. Oxford University Press (1953)
2. Civril, A., Magdon-Ismail, M.: On Selecting A Maximum Volume Sub-matrix of a Matrix and Related Problems. TCS 410(47-49), 4801–4811 (2009)
3. Cutler, A., Breiman, L.: Archetypal Analysis. Technometr. 36(4), 338–347 (1994)
4. Kersting, K., Wahabzada, M., Roemer, C., Thurau, C., Ballvora, A., Rascher, U., Leon, J., Bauckhage, C., Pluemer, L.: Simplex distributions for embedding data matrices over time. In: SDM (2012)
5. Kersting, K., Xu, Z., Wahabzada, M., Bauckhage, C., Thurau, C., Roemer, C., Ballvora, A., Rascher, U., Leon, J., Pluemer, L.: Pre–symptomatic prediction of plant drought stress using dirichlet–aggregation regression on hyperspectral images. In: AAAI — Computational Sustainability and AI Track (2012)
6. Mahoney, M.W., Drineas, P.: CUR Matrix Decompositions for Improved Data Analysis. PNAS 106(3), 697–702 (2009)
7. Thurau, C., Kersting, K., Bauckhage, C.: Yes We Can – Simplex Volume Maximization for Descriptive Web-Scale Matrix Factorization. In: Proc. CIKM (2010)
8. Thurau, C., Kersting, K., Bauckhage, C.: Deterministic CUR for improved large–scale data analysis: An empirical study. In: SDM (2012)
9. Thurau, C., Kersting, K., Wahabzada, M., Bauckhage, C.: Descriptive matrix factorization for sustainability: Adopting the principle of opposites. DAMI 24(2), 325–354 (2012)
10. Torralba, A., Fergus, R., Freeman, W.T.: 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence 30(11), 1958–1970 (2008)

Metal Binding in Proteins: Machine Learning Complements X-Ray Absorption Spectroscopy

Marco Lippi[1], Andrea Passerini[2], Marco Punta[3], and Paolo Frasconi[4]

[1] Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy
[2] Dipartimento di Ingegneria e Scienza dell'Informazione, Università di Trento, Italy
[3] Wellcome Trust Sanger Institute, Hinxton, UK
[4] Dipartimento di Sistemi e Informatica, Università di Firenze, Italy

Abstract. We present an application of machine learning algorithms for the identification of metalloproteins and metal binding sites on a genome scale. An extensive evaluation conducted in combination with X-ray absorption spectroscopy shows the great potentiality of the approach.

1 Metal Binding in Proteins

A significant fraction of known proteins is believed to bind metal ions in their native conformation. Metal ions play a variety of crucial roles in proteins [1], from stabilizing their three dimensional structure, to acting as cofactors in enzyme catalysis. Moreover, metals are implicated in many diseases for which medicine is still seeking an effective treatment, such as Parkinson's or Alzheimer's [2] . Identifying unknown metalloproteins and detecting their metal binding site(s) is an important step in understanding their function and characterizing many crucial processes involved in living systems. A metal binding site is characterized through its metal ion, the protein amino acid residues directly involved in binding it (called ligands) and the binding geometry, i.e. the spatial arrangement of the ion and its ligands. Some metal binding sites actually involve compounds (e.g. the heme group binding hemoglobin) including one or more ions, and some proteins contain more than one metal binding site. The problem of identifying and characterizing metalloproteins can be seen as a series of increasingly complex tasks. Given a protein, one aims at: 1) determining whether it is a metalloprotein or not, i.e. if it binds metal ions in its native conformation; 2) determining the metal bonding state of each of its residues, i.e. whether they bind a metal ion or not; 3) determining the composition of metal binding sites, i.e. the number of ions binding the protein and the set of their respective ligands. Answers may be obtained experimentally, by means of in-silico prediction tools, or by a combination of the two classes of methods.

Experimental Methods. High-throughput techniques based on X-ray absorption spectroscopy [3] (HT-XAS) allow detection, identification and quantification of metals bound to proteins based on the energy and intensity of the X-ray fluorescence signal emitted by the metals. HT-XAS can thus be used to address the first task, i.e. metalloprotein identification. Exact determination of binding sites, however, is only possible using more labor-intensive techniques, such as X-ray

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 854–857, 2012.

crystallography or Nuclear Magnetic Resonance, which provide high-resolution three-dimensional structural information. Even when a 3D structure is available, exact characterization of binding sites may be non trivial and error prone. For example false positives may be due to spurious artifacts where metals bind at adventitious sites, and false negatives may emerge when metalloproteins are experimentally solved in their apo-form lacking the metal ion.

In Silico Methods. The three above tasks can be tackled from a machine learning point of view. Here we focus on predictions from sequence alone[1], where (1) is a sequence classification problem, (2) is a sequence labeling problem, and (3) is a more complex structured output problem from sequences to bipartite graphs. Some simplifying assumptions may be made to reduce the difficulty of these problems in their generality. First, prediction may be limited to transition metals and a small number of candidate residues (CYS and HIS). Transition metals (especially iron and zinc) are the most commonly found ions in proteins, covering about 2/3 of all known metalloproteins. Their preferred ligands are CYS and HIS, followed by ASP and GLU, which have a much lower binding propensity given their relatively high abundance in proteins. Finally, the solution space may be limited to sites where an ion is coordinated by four or less residues as more complex binding sites are extremely rare.

MetalDetector. The MetalDetector software [5,6] uses state-of-the-art machine learning methods to solve the above three prediction problems. A first version of the software [5] employed Disulfind predictor to identify cysteine disulfide bridges [7] and a combination of support vector machines and bidirectional recurrent neural networks for metal bonding state prediction. The current version[2] of the server [6] employs a two-stage approach for metal bonding state and metal binding sites prediction respectively. The first stage relies on an SVM-HMM [8] which collectively assigns the bonding state of all the CYS/HIS residues in the sequence. Residues predicted as metal-bound are fed to the second stage. Here a search-based structure output approach greedily adds links between candidate ligands and candidate ions, until each ligand is connected to an ion. The search is guided by a kernel-based scoring function trained to score correct moves higher than incorrect moves. The problem has the structure of a weighted matroid, which is basically the discrete counterpart of concave functions. The greedy search is thus guaranteed to lead to the global optimum of the (learned) scoring function. The method was initially introduced in [9] and further refined and analyzed in [10], where an extensive experimental validation across different protein structural folds and superfamilies was conducted.

Being able to address all three predictions problems, MetalDetector is a natural candidate to complement information provided by high-throughput experimental techniques like HT-XAS. The potential impact of this integration was recently shown on a large-scale experiment aimed at identifying potential metalloproteins within the New York SGX Research Center for Structural Genomics.

[1] For applications of machine learning techniques to 3D structure data see e.g. [4].

[2] MetalDetector is available as a web server at `http://metaldetector.dsi.unifi.it`.

2 Results

MetalDetector and HT-XAS were jointly employed in a recent study [11] in order to identify metal-bonded residues in 3,879 purified proteins generated by the New York SGX Research Center for Structural Genomics and belonging to hundreds of different protein families.

Of the whole set of proteins, 343 were identified by HT-XAS to contain at least one metal ion among Mn, Fe, Co, Ni, Cu and Zn. The experimental analysis described in [11] compares the level of agreement between MetalDetector and HT-XAS predictions: to this aim, MetalDetector predictions at residue level have been combined in order to define a protein score, which is used to predict whether that protein is a metalloprotein or not. This level of agreement obviously depends on the aggregation criterion used to produce such protein scores for MetalDetector: in these experiments, the adopted criterion was to predict a protein to be a metalloprotein if for at least N residues (either CYS or HIS) the probability of metal bonding state, as predicted by MetalDetector, exceeded a certain threshold T_M. By choosing different values for N and T_M, different predictions can be accordingly obtained: the experiments showed that, at the same recall level (i.e., when MetalDetector predicts the same number of metalloproteins as HT-XAS), MetalDetector and HT-XAS agree from 32% up to 45% of the cases, depending on the choice of N and T_M (note that a random baseline predictor would achieve a 10% of precision with respect to the metalloproteins identified by HT-XAS).

In addition, it must be underlined that in many protein samples metal occupancy can be low, and therefore metal atoms cannot be detected by HT-XAS. MetalDetector can in these cases complement HT-XAS evidence by suggesting potentially missed metalloproteins. This happens, for example, for proteins 11211f and 11213j, which share the Pfam SCO1/SenC domain (PF02630) involved in biogenesis of respiratory and photosynthetic systems. Protein 11211f shares 26% sequence identity with Human SCO2 protein, a mitochondrial membrane-bound protein involved in copper supply for the assembly of cytochrome c oxidase. The residues predicted by MetalDetector to bind metal in 11211f align to the residues that in the NMR structure of Human SCO2 bind a Cu^+ ion. This is likely one of the cases where the HT-XAS method fails in identifying a metalloprotein, while MetalDetector not only seems to recover the false negative, but also to correctly predict the position of the binding site.

A comprehensive approach combining the use of MetalDetector predictions with homology modeling has also been object of analysis and showed that the proposed computational methodology can represent an extremely powerful tool for the study of metalloproteins.

3 Perspectives

Recent years have witnessed a dramatic increase in the availability of high-throughput experimental techniques for the analysis of biological data. This scenario provides an unprecedented opportunity for machine learning approaches to deal with large amount of data and continuously novel problems and challenges. Structural genomics, which aims to map the protein sequence space with structural information, is indeed pursuing a tight integration of high-throughput experimental techniques and modeling approaches. Characterization of metalloproteins is an interesting example of how experimental techniques and machine learning approaches can be fruitfully combined to deepen our understanding of biological systems.

Acknowledgments. ML, AP and PF were partially supported by grant PRIN 2009LNP494 (Statistical Relational Learning: Algorithms and Applications) from Italian Ministry of University and Research. MP is supported by Wellcome Trust (grant numbers WT077044/Z/05/Z). We gratefully acknowledge the collaboration with W. Shi, J. Bohon, M. Sauder, R. D'Mello, M. Sullivan, J. Toomey, D. Abel, S.K. Burley, B. Rost and M.R. Chance.

References

1. Bertini, I., Sigel, A., Sigel, H.: Handbook on Metalloproteins. M. Dekker (2001)
2. Barnham, K.J., Bush, A.I.: Metals in Alzheimer's and Parkinson's diseases. Current Opinion in Chemical Biology 12(2), 222–228 (2008)
3. Shi, W., Zhan, C., Ignatov, A., Manjasetty, B.A., Marinkovic, N., Sullivan, M., Huang, R., Chance, M.R.: Metalloproteomics: High-throughput structural and functional annotation of proteins in structural genomics. Structure 13(10), 1473–1486 (2005)
4. Babor, M., Gerzon, S., Raveh, B., Sobolev, V., Edelman, M.: Prediction of transition metal-binding sites from apo protein structures. Proteins 70(1), 208–217 (2007)
5. Lippi, M., Passerini, A., Punta, M., Rost, B., Frasconi, P.: Metaldetector: a web server for predicting metal-binding sites and disulfide bridges in proteins from sequence. Bioinformatics 24(18), 2094–2095 (2008)
6. Passerini, A., Lippi, M., Frasconi, P.: Metaldetector v2.0: predicting the geometry of metal binding sites from protein sequence. Nucl. Ac. Res. 39(Web-Server-Issue), 288–292 (2011)
7. Ceroni, A., Passerini, A., Vullo, A., Frasconi, P.: Disulfind: a disulfide bonding state and cysteine connectivity prediction server. Nucl. Ac. Res. 34, 177–181 (2006)
8. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. 6, 1453–1484 (2005)
9. Frasconi, P., Passerini, A.: Predicting the geometry of metal binding sites from protein sequence. In: NIPS 21, Vancouver, Canada, pp. 465–472. MIT Press (2008)
10. Passerini, A., Lippi, M., Frasconi, P.: Predicting metal-binding sites from protein sequence. IEEE/ACM Trans. Comput. Biology Bioinform. 9(1), 203–213 (2012)
11. Shi, W., Punta, M., Bohon, J., Sauder, J.M., D'Mello, R., Sullivan, M., Toomey, J., Abel, D., Lippi, M., Passerini, A., Frasconi, P., Burley, S.K., Rost, B., Chance, M.R.: Characterization of metalloproteins by high-throughput x-ray absorption spectroscopy. Genome Research 21(6), 898–907 (2011)

Modelling Input Varying Correlations between Multiple Responses

Andrew Gordon Wilson and Zoubin Ghahramani

University of Cambridge,
Cambridge, UK
agw38@cam.ac.uk, zoubin@eng.cam.ac.uk

Abstract. We introduced a generalised Wishart process (GWP) for modelling input dependent covariance matrices $\Sigma(x)$, allowing one to model input varying correlations and uncertainties between multiple response variables. The GWP can naturally scale to thousands of response variables, as opposed to competing *multivariate volatility* models which are typically intractable for greater than 5 response variables. The GWP can also naturally capture a rich class of covariance dynamics – periodicity, Brownian motion, smoothness, ... – through a covariance kernel.

1 Introduction

Modelling covariances between random variables is fundamental in statistics. For convenience, covariances between multiple responses are usually assumed to be constant. However, accounting for how these covariances depend on inputs (e.g. time) can greatly improve statistical inferences. For example, to predict the expression level of a gene at a particular time, it helps to consider the expression levels of correlated genes, and how these correlations depend on time.

Modelling of dependent covariances between multiple responses is largely uncharted territory. The small number of existing models for dependent covariances are mostly found in the econometrics literature, and are referred to as *multivariate volatility models*. In econometrics, a good estimate of a time varying covariance matrix $\Sigma(t) = \text{cov}[\boldsymbol{r}(t)]$ for a vector of returns $\boldsymbol{r}(t)$ is useful for estimating the risk of a particular portfolio. Multivariate volatility models are also used to understand *contagion*: the transmission of a financial shock from one entity to another (Bae et al., 2003). However, it is generally useful – in econometrics, machine learning, or otherwise – to know input dependent uncertainty, and the dynamic correlations between multiple entities.

Despite their importance, conventional multivariate volatility models suffer from tractability issues and a lack of generality. MGARCH (Bollerslev et al., 1988; Silvennoinen and Teräsvirta, 2009), multivariate stochastic volatility (Harvey et al., 1994; Asai et al., 2006), and the original Wishart process (Bru, 1991; Gouriéroux et al., 2009), are typically highly parametrised, parameters are often difficult to interpret or estimate (given the constraint $\Sigma(t)$ must be positive definite), are typically intractable for more than 5 response variables, and are

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 858–861, 2012.

restricted to Brownian motion or Markovian covariance dynamics (Silvennoinen and Teräsvirta, 2009; Gouriéroux, 1997; Gouriéroux et al., 2009).

Modelling of dependent covariances is beautifully suited to a Bayesian nonparametric approach. We introduced the Bayesian nonparametric *generalised Wishart process* (GWP) prior (Wilson and Ghahramani, 2010, 2011) over input dependent matrices $\Sigma(x)$, where $x \in \mathcal{X}$ is an arbitrary input variable. The generalised Wishart process volatility model has the following desirable properties:

1. The GWP is tractable for up to at least 1000×1000 covariance matrices.
2. The small number of free parameters give information about the underlying source of volatility, like whether there is periodicity (and if so what the period would be), and how far into the past one should look for good forecasts.
3. The input variable can be any arbitrary $x \in \mathcal{X}$ just as easily as it can represent time (useful for spatially varying dependencies, and for including covariates like interest rates in time series models).
4. The dynamics of $\Sigma(x)$ can easily be specified as periodic, smooth, Brownian motion, etc., through a kernel function.
5. Missing data are handled easily, and there is prior support for any (uncountably infinite) sequence of covariance matrices $\{\Sigma(x_1), \ldots, \Sigma(x_n)\}$.

2 Construction

The Wishart distribution is a distribution over positive definite matrices. Given a $p \times \nu$ matrix A with entries $A_{ij} \sim \mathcal{N}(0,1)$, and a lower triangular matrix of constants L, the product $LAA^\top L^\top$ has a Wishart distribution:

$$LAA^\top L^\top \sim \mathcal{W}_p(\nu, LL^\top)\,. \tag{1}$$

To turn the Wishart distribution into a generalised Wishart process (in its simplest form), one replaces the Gaussian random variables with *Gaussian processes* (Rasmussen and Williams, 2006). We let the matrix A be a function of inputs x, by filling each entry with a Gaussian process: $A_{ij}(x) \sim \mathcal{GP}(0,k)$. We let

$$\Sigma(x) = LA(x)A(x)^\top L^\top\,. \tag{2}$$

At any given x, the matrix $A(x)$ is a matrix of Gaussian random variables, since a Gaussian process function evaluated at any input location is simply a Gaussian random variable. Therefore at any x, $\Sigma(x)$ has a Wishart marginal distribution. $\Sigma(x)$ is a collection of positive definite matrices, indexed by x, and dynamics controlled by the covariance kernel k. $\Sigma(x)$ has a generalised Wishart process prior, and we write $\Sigma(x) \sim \mathcal{GWP}(\nu, L, k)$. The parameters are easily interpretable. L controls the prior expectation of $\Sigma(x)$ at any x: $\mathbb{E}[\Sigma(x)] = \nu LL^\top$. The greater ν the greater our confidence in this prior expectation. The covariance kernel controls how the entries of $\Sigma(x)$ vary with x: $\mathrm{cov}(\Sigma_{ij}(x), \Sigma_{ij}(x')) \propto k(x,x')^2$. A single draw from a GWP prior over 2×2 covariance matrices is illustrated in Figure 1. Given vector valued observations $\boldsymbol{r}(x)$ (e.g. a vector of stock returns indexed by x), we can efficiently infer a posterior over the generalised Wishart process using Elliptical Slice Sampling (Murray et al., 2010), which is a recent MCMC technique designed to sample from posteriors with Gaussian priors.

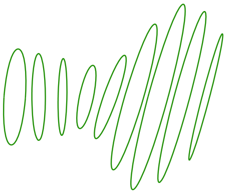


Fig. 1. A draw from a generalised Wishart process (GWP). Each ellipse is a 2×2 covariance matrix indexed by time, which increases from left to right. The rotation indicates the correlation between the two variables, and the axes scale with the diagonals of the matrix. Like a draw from a Gaussian process is a collection of function values indexed by time, a draw from a GWP is a collection of matrices indexed by time.

3 Results

We generated a 2×2 time varying covariance matrix $\Sigma_p(t)$ with periodic components, simulating data at 291 time steps from a Gaussian:

$$\boldsymbol{y}(t) \sim \mathcal{N}(\mathbf{0}, \Sigma_p(t)) . \tag{3}$$

Periodicity is especially common to financial and climate data, where daily trends repeat themselves. For example, the intraday volatility on equity indices and currency exchanges has a periodic covariance structure. Andersen and Bollerslev (1997) discuss the lack of – and critical need for – models that account for this periodicity. With a GWP, we can simply use a periodic kernel function, whereas in previous Wishart process volatility models (Bru, 1991; Gouriéroux et al., 2009), we are stuck with a Markovian covariance structure. Figure 2 shows the results. We also performed step ahead forecasts of $\Sigma(t)$ on financial data with promising results, elucidated in Wilson and Ghahramani (2010, 2011). The recent *Gaussian process regression network* (GPRN) (Wilson et al., 2011, 2012) uses a GWP noise model, and extends the multi-task Gaussian process framework to handle *input dependent* signal and noise correlations between multiple responses. The GPRN has strong predictive performance and scalability on many real datasets, including a gene expression dataset with 1000 response variables.

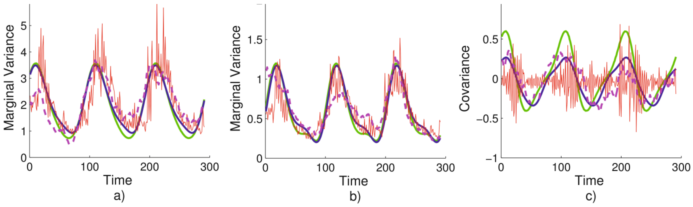


Fig. 2. Reconstructing the historical $\Sigma_p(t)$ for the periodic data set. We show the truth (green), and GWP (blue), WP (dashed magenta), and MGARCH (thin red) predictions. a) and b) are the diagonal elements of $\Sigma_p(t)$, c) is the covariance.

References

Andersen, T.G., Bollerslev, T.: Intraday periodicity and volatility persistence in financial markets. Journal of Empirical Finance 4(2-3), 115–158 (1997)

Asai, M., McAleer, M., Yu, J.: Multivariate stochastic volatility: a review. Econometric Reviews 25(2), 145–175 (2006)

Bae, K., Karolyi, G., Stulz, R.: A new approach to measuring financial contagion. Review of Financial Studies 16(3), 717 (2003)

Bollerslev, T., Engle, R.F., Wooldridge, J.M.: A capital asset pricing model with time-varying covariances. The Journal of Political Economy 96(1), 116–131 (1988)

Bru, M.: Wishart processes. Journal of Theoretical Probability 4(4), 725–751 (1991)

Gelfand, A., Schmidt, A., Banerjee, S., Sirmans, C.: Nonstationary multivariate process modeling through spatially varying coregionalization. Test 13(2), 263–312 (2004)

Gouriéroux, C.: ARCH models and financial applications. Springer (1997)

Gouriéroux, C., Jasiak, J., Sufana, R.: The Wishart autoregressive process of multivariate stochastic volatility. Journal of Econometrics 150(2), 167–181 (2009)

Harvey, A., Ruiz, E., Shephard, N.: Multivariate stochastic variance models. The Review of Economic Studies 61(2), 247–264 (1994)

Murray, I., Adams, R.P., MacKay, D.J.: Elliptical Slice Sampling. JMLR: W&CP 9, 541–548 (2010)

Rasmussen, C.E., Williams, C.K.: Gaussian processes for Machine Learning. The MIT Press (2006)

Silvennoinen, A., Teräsvirta, T.: Multivariate GARCH models. Handbook of Financial Time Series, pp. 201–229 (2009)

Wilson, A.G., Knowles, D.A., Ghahramani, Z.: Gaussian process regression networks. Arxiv preprint arXiv:1110.4411 (2011)

Wilson, A.G., Knowles, D.A., Ghahramani, Z.: Gaussian process regression networks. In: International Conference on Machine Learning (2012)

Wilson, A.G., Ghahramani, Z.: Generalised Wishart Processes. Arxiv preprint arXiv:1101.0240 (2010)

Wilson, A.G., Ghahramani, Z.: Generalised Wishart Processes. In: UAI (2011)

Author Index